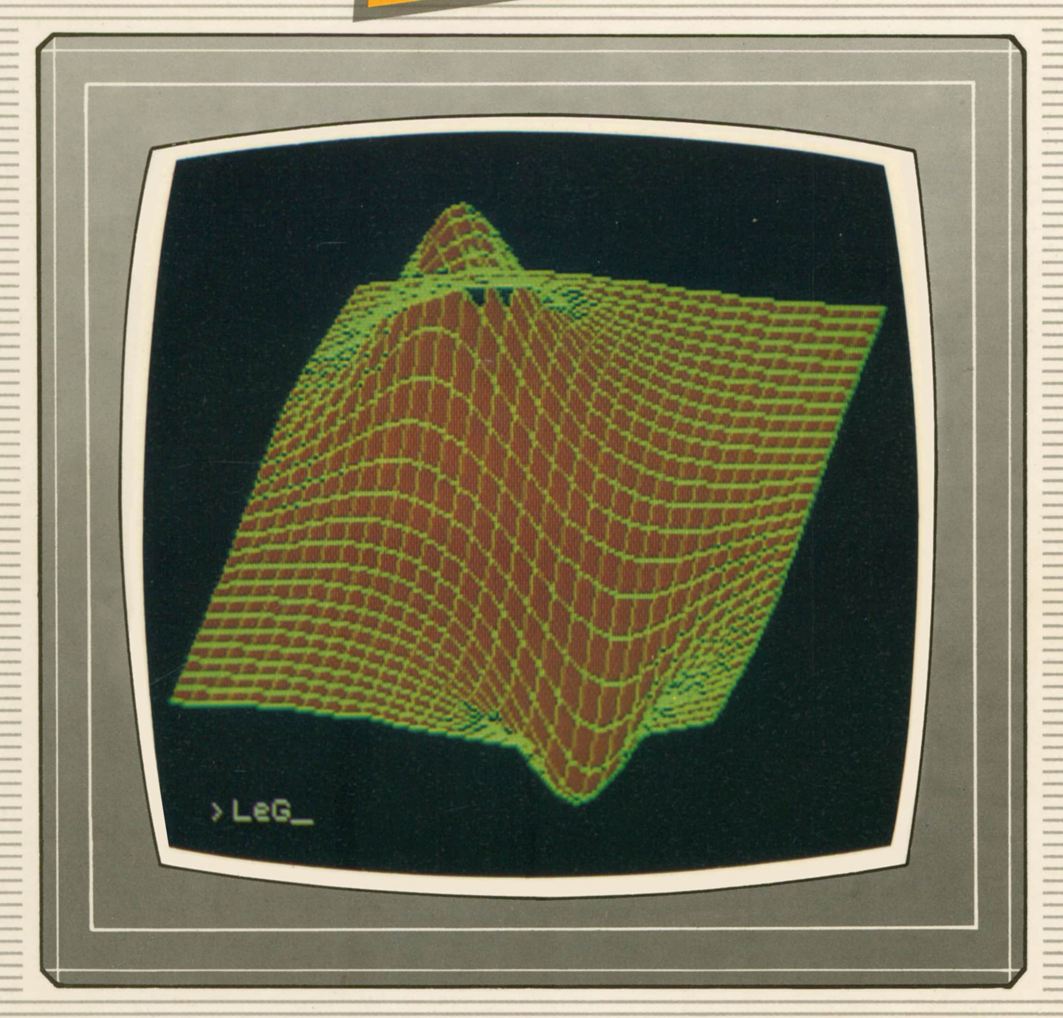
▼ AMSTRAD ▼ SPECTRUM ▼ COMMODORE ▼ IBM ▼ MSX ▼

Informática y programación 31

PASO A PASO



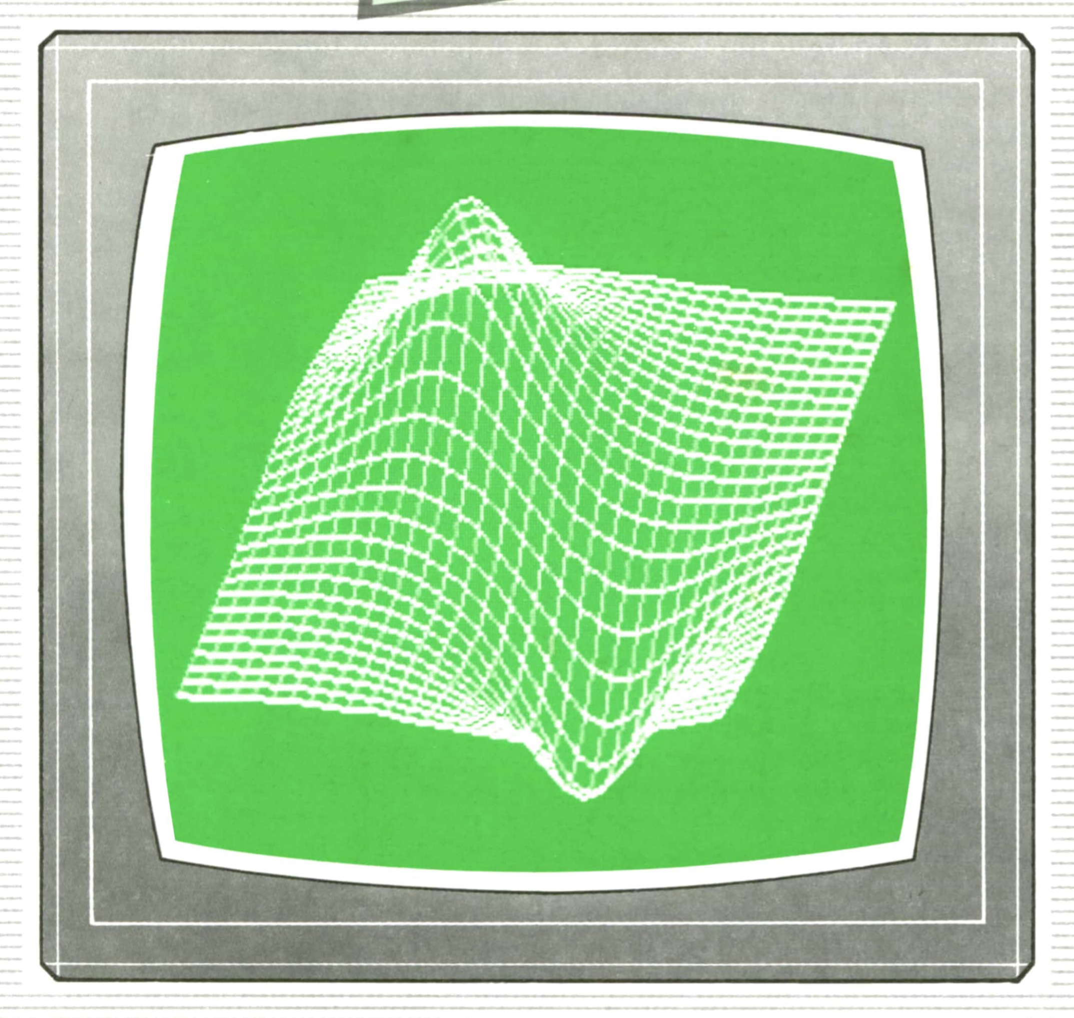
PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Informática y programación 31

PASO A PASO



PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

▼ BASIC ▼ MAQUINA ▼ PASCAL ▼ LOGO ▼ OTROS LENGUAJES ▼
▼ TECNICAS DE ANALISIS Y DE PROGRAMACION ▼

▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼

Una publicación de

EDICIONES SIGLO CULTURAL, S.A.

Director-editor:
RICARDO ESPAÑOL CRESPO.

Gerente:
ANTONIO G. CUERPO.

Directora de producción:
MARIA LUISA SUAREZ PEREZ.

Directores de la colección:
MANUEL ALFONSECA, Doctor Ingeniero de Telecomunicación y Licenciado en Informática.
JOSE ARTECHE, Ingeniero de Telecomunicación.

Diseño y maquetación:
BRAVO-LOFISH.

Fotografía:
EQUIPO GALATA.

Dibujos:
JOSE OCHOA

TECNICAS DE PROGRAMACION: Manuel Alfonseca, Doctor Ingeniero de Telecomunicación y Licenciado en Informática. TECNICAS DE ANALISIS: José Arteche, Ingeniero en Telecomunicación. LENGUAJE MAQUINA 8086: Juan Rojas Licenciado en Ciencias Físicas e Ingeniero Industrial. PASCAL: Juan Ignacio Puyol, Ingeniero Industrial. PROGRAMAS (educativos, de utilidad, de gestión y de juegos): Francisco Morales, Técnico en Informática y colaboradores. Coordinador de AULA DE INFORMATICA APLICADA (AIA): Alejandro Marcos, Licenciado en Ciencias Químicas. BASIC: Esther Maldonado, Diplomada en Arquitectura. INFORMATICA BASICA: Virginia Muñoz, Diplomada en Informática. LENGUAJE MAQUINA Z-80: Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Tojo, Diplomado en Telecomunicación. LENGUAJE MAQUINA 6502: (desde el tomo 5): Juan José Gómez, Licenciado en Química. LOGO: Cristina Manzanera, Licenciada en Informática. APLICACIONES: Jorge Thomas, Técnico en Informática. OTROS LENGUAJES (ADA): Joaquín Salvachúa, Diplomado en Telecomunicación y José Luis Tojo, Diplomado en Telecomunicación.

Ediciones Siglo Cultural, S.A.

Dirección, redacción y administración:
Pedro Teixeira, 8, 2.ª planta. Teléf. 255 09 99. 28020 Madrid.

Publicidad:
Gofar Publicidad, S.A. Benito de Castro, 12 bis. 28028 Madrid.

Distribución en España:
COEDIS, S.A. Valencia, 245. Teléf. 215 70 97. 08007 Barcelona.
Delegación en Madrid: Serrano, 165. Teléf. 411 11 48.

Distribución en Ecuador: Muñoz Hnos.

Distribución en Perú: DISELPESA.

Distribución en Chile: Alfa Ltda.

Importador exclusivo Cono Sur:
CADE, S.R.L. Pasaje Sud América, 1532. Teléf.: 21 24 64.
Buenos Aires - 1.290. Argentina.

ISBN del tomo: 84-7688-182-7
ISBN de la obra: 84-7688-068-7

Fotocomposición:
ARTECOMP, S.A. Albarracín, 50. 28037 Madrid.

Imprime:
MATEU CROMO. Pinto (Madrid).

Depósito legal: M-5-677-1987
Printed in Spain - Impreso en España.

Suscripciones y números atrasados:
Ediciones Siglo Cultural, S.A.
Pedro Teixeira, 8, 2.ª planta. Teléf. 259 73 31. 28020 Madrid.

Enero, 1988
P.V.P. Canarias: 335,–.

INDICE

4 INFORMATICA BASICA

8 MAQUINA 8088

11 PROGRAMAS EDUCATIVOS
PROGRAMAS DE UTILIDAD
PROGRAMAS DE GESTION
PROGRAMAS DE JUEGOS

26 TECNICAS DE ANALISIS

28 TECNICAS DE PROGRAMACION

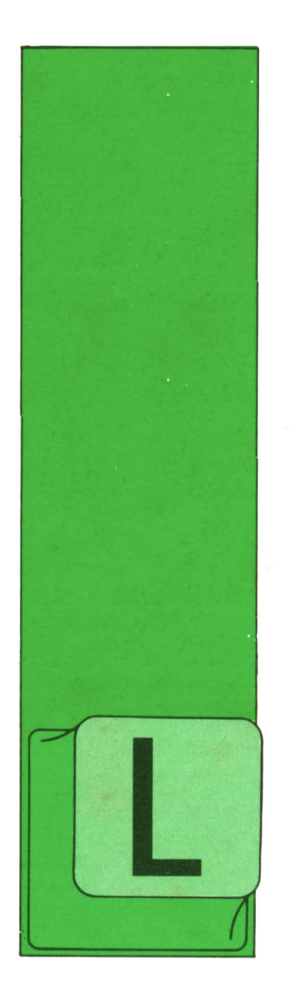
32 LOGO

35 PASCAL

39 OTROS LENGUAJES

BASIC

MATRICES (II)





Matrices bidimensionales

LAS matrices bidimensionales son agrupaciones de variables simples en dos dimensiones, es decir, las variables se ordenan en filas y columnas formando una tabla, tal y como se muestra en la figura 1.

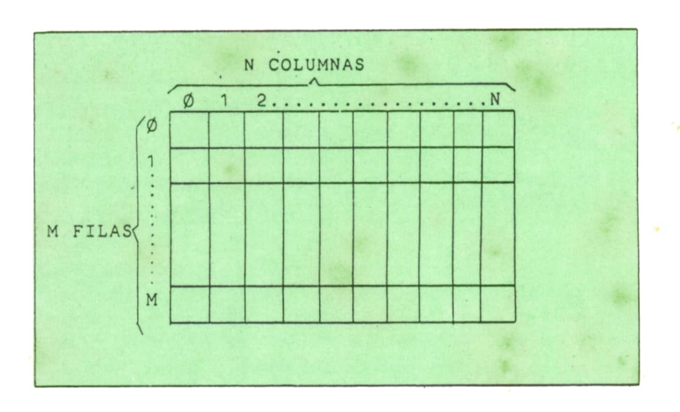




Fig. 1. Representación esquemática de una matriz bidimensional. Cada recuadro representa una variable o posición de matriz.

En una matriz bidimensional cada posición viene especificada por dos parámetros: el primero indica el número de la fila y el segundo el número de la columna. Por tanto, para hacer referencia a una variable de una matriz de dos dimensiones tendremos que escribir el nombre de la variable correspondiente a dicha matriz, y, a continuación, dos números separados por una coma. Por ejemplo:

J (8,2)

hace referencia al elemento de la matriz J (numérica) que está situado en la fila 8 y en la columna 2.

Al igual que sucedía con las matrices de una dimensión, tanto las filas como las columnas se empiezan a numerar por cero. La excepción la presenta el SPECTRUM, que empieza a numerar por uno.

Supongamos una matriz A de 3 filas y 2 columnas. El nombre de cada variable elemental está indicado en la figura 2.

A(Ø,Ø)	A(Ø,1)
A(1,Ø)	A(1,1)
A(2,Ø)	A(2,1)



Fig. 2. Elementos de una matriz de 3 filas y 2 columnas.

Sin embargo, en el SPECTRUM el nombre de cada elemento sigue las pautas representadas en la figura 3.

Para trabajar con matrices de dos dimensiones es necesario dimensionarlas primero. Para ello se utiliza, como ya sabemos, la instrucción DIM, sólo que ahora con el siguiente formato:

DIM < nombre de variable > (n.° de filas, n.° de columnas – 1)

A(1,1)	A(1,2)
A(2,1)	A(2,2)
A(3,1)	A(3,2)



Fig. 3. Elementos de una matriz de 3 filas y 2 columnas en el SPECTRUM.

En el SPECTRUM el formato será:
DIM < nombre de variable >
(n.º de filas, n. º de columnas)

Recordemos que cuando trabajamos con matrices unidimensionales utilizábamos un bucle FOR-NEXT cuya variable índice se utiliza como subíndice para los elementos de la matriz. Ahora necesitamos dos subíndices, por lo que tendremos que utilizar dos bucles anidados.

Vemos un ejemplo. El programa 1 carga una matriz de F filas y C columnas con números enteros al azar comprendidos entre 0 y 99, y a continuación imprime la matriz en pantalla, en forma de tabla.

```
10 REM **************************
20 REM *  MATRIZ BIDIMENSIONAL  *
30 REM **************************
40 CLS
50 INPUT "NUMERO DE FILAS ";F
60 INPUT "NUMERO DE COLUMNAS ";C
70 CLS
80 DIM A(F,C)
90 FOR I=1 TO F
100 FOR J=1 TO C
110 LET A(I,J)=INT(RND(1)*100)
120 NEXT J
130 NEXT I
140 LET N=INT(40/C)
150 FOR I=1 TO F
160 LET T=1
170 FOR J=1 TO C
180 PRINT TAB(T);A(I,J);
190 LET T=T+N
200 NEXT J
210 NEXT I
```

En la figura 4 podemos ver la evolución del estado de la matriz A del programa 1 suponiendo que al ejecutarlo asignamos el valor 3 a la variable F y el valor 2 a la variable C (matriz de 3 filas y 2 columnas).

Nº DE VUELTA DEL BUCLE		POSICION ASIGNADA	ESTADO DE LA MATRIZ
I	J	A (I,J)	
1	1	A (1,1)	
1	2	A (1,2)	
2	1	A (2,1)	
2	2	A (2,2)	
3	1	A (3,1)	
3	2	A (3,2)	



Fig. 4. Evolución del programa 1 en el proceso de carga de datos en la matriz para F = 3 y C = 2.

Podemos observar que en el programa 1 hemos desechado la fila 0 y la columna 0 de la matriz. Esto tiene como único objeto que los bucles de carga de datos e impresión de resultados puedan servir para todos los ordenadores, incluso el SPECTRUM.

La impresión en pantalla se ha pensado para una pantalla de 40 columnas. Si la pantalla es de 32 columnas, habrá que sustituir la línea 140 por:

140 LET N = INT (32/C)

y si es de 80 columnas:

140 LET N = INT (80/C)

Finalmente, en la figura 5 podemos ver un ejemplo de la ejecución del programa 1, mostrando en pantalla una matriz de 20 filas y 8 columnas.

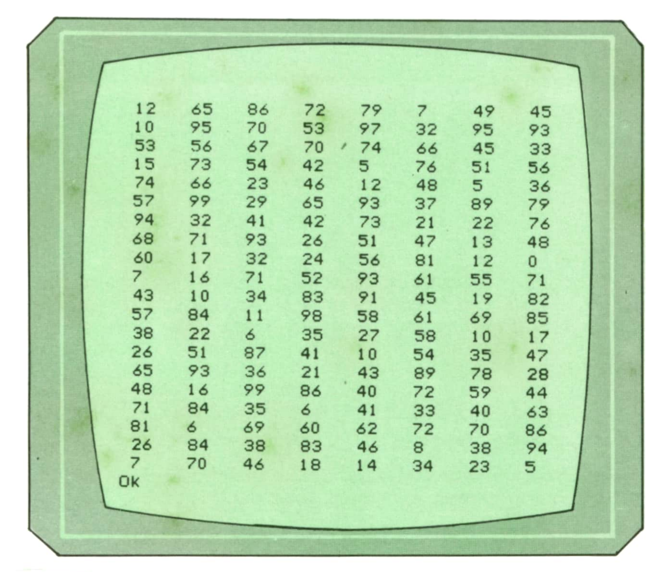




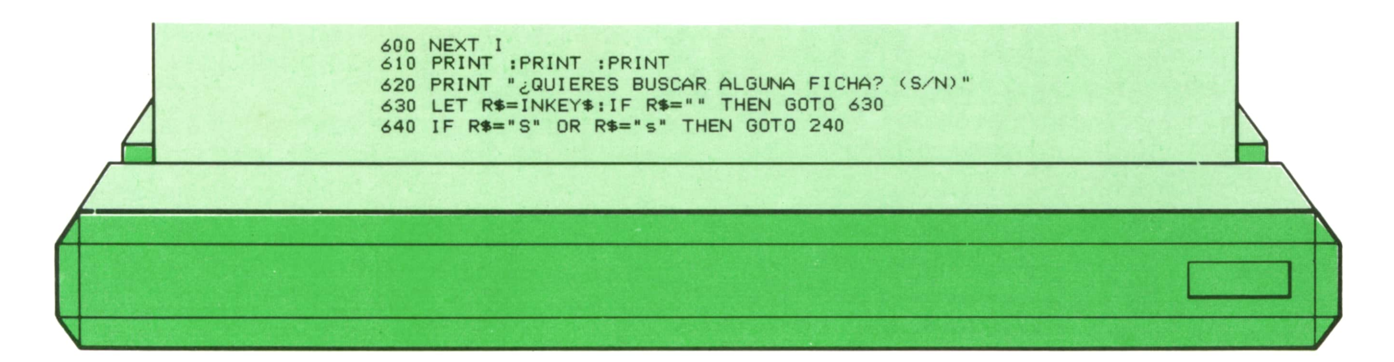
Fig. 5. Presentación en pantalla del programa 1.

En cuanto a las matrices alfanuméricas bidimensionales, se trabaja exactamente igual que con las numéricas. En el SPECTRUM como sucedía con las matrices alfanuméricas unidimensionales, hay que indicar además la longitud máxima de las cadenas que se van a almacenar en la matriz.

Por tanto, si escribimos DIM N$ (8,5,12) en el SPECTRUM, estamos reservando memoria para una matriz N$ de 8 filas y 5 columnas con 12 caracteres como máximo de longitud de cada cadena.

En el programa 2 tenemos un ejemplo de utilización de matrices alfanuméricas bidimensionales para la creación de un fichero de datos compuesto por F fichas y C campos por ficha. Además, el mismo programa nos permite buscar una ficha cualquiera por el campo que deseemos. Para ello, la fila 0 de la matriz almacena los nombres de los campos.

```
10 REM **********************
20 REM *  FICHERO DE DATOS  *
30 REM **********************
40 CLS
50 INPUT "NUMERO DE FICHAS";F
60 INPUT "NUMERO DE CAMPOS POR FICHA";C
70 CLS
80 DIM N$(F,C)
90 FOR I=1 TO C
100 PRINT "NOMBRE DEL CAMPO ";I;
110 INPUT N$(0,I)
120 NEXT I
130 CLS
140 FOR I=1 TO F
150 PRINT "FICHA ";I
160 FOR J=1 TO C
170 PRINT N$(0,J);
180 INPUT N$(I,J)
190 NEXT J
200 CLS
210 NEXT I
220 PRINT "YA ESTA CREADO EL FICHERO"
230 GOTO 610
240 CLS
250 PRINT :PRINT
260 PRINT TAB(17);"OPCIONES"
270 PRINT TAB(17);"________"
280 PRINT :PRINT
290 PRINT TAB(16);"0 . NUMERO DE FICHA"
300 PRINT
310 FOR I=1 TO C
320 PRINT TAB(15);I;". ";N$(0,I)
330 PRINT
340 NEXT I
350 PRINT :PRINT :PRINT
360 INPUT "SELECCIONA EL CAMPO POR EL QUE DESEAS BUSCAR";R
370 IF R<0 OR R>C  THEN GOTO 360
380 CLS
390 IF R=0 THEN GOTO 530
400 PRINT N$(0,R);" BUSCADO";
410 INPUT B$
420 FOR I=1 TO F
430 IF N$(I,R)=B$ THEN GOTO 460
440 NEXT I
450 PRINT B$;" NO FIGURA EN EL FICHERO":GOTO 610
460 CLS
470 PRINT "FICHA ";I
480 FOR J=1 TO C
490 PRINT
500 PRINT N$(0,J);":",N$(I,J)
510 NEXT J
520 GOTO 610
530 INPUT "NUMERO DE FICHA BUSCADO";N
540 IF N<1 OR N>F THEN GOTO 530
550 CLS
560 PRINT "FICHA ";N
570 FOR I=1 TO C
580 PRINT
590 PRINT N$(0,I);":",N$(N,I)
```



En las figuras 6 y 7 podemos ver distintos momentos de la ejecución del programa 2.

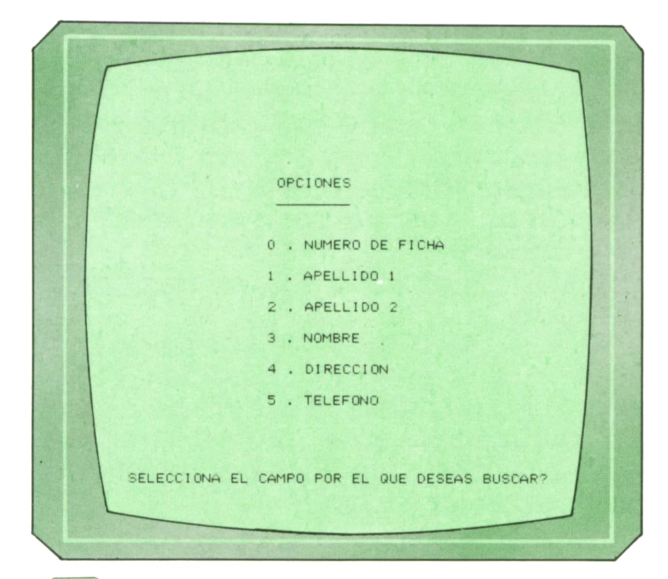




Fig. 6. Opciones de búsqueda de una ficha en el programa 2.

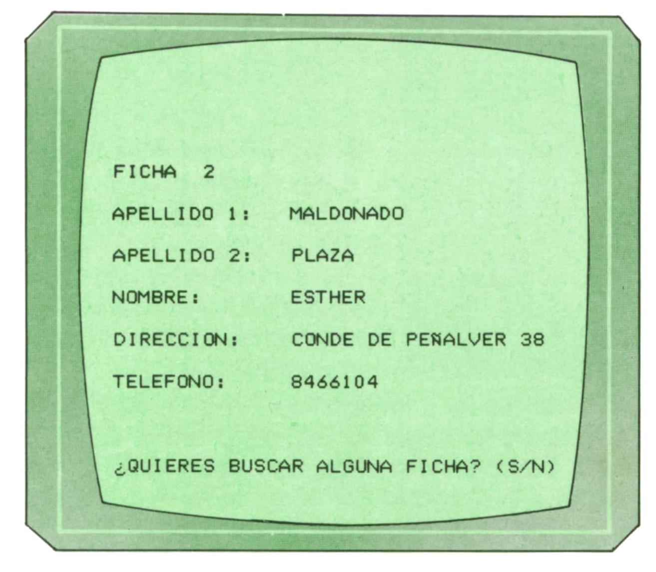
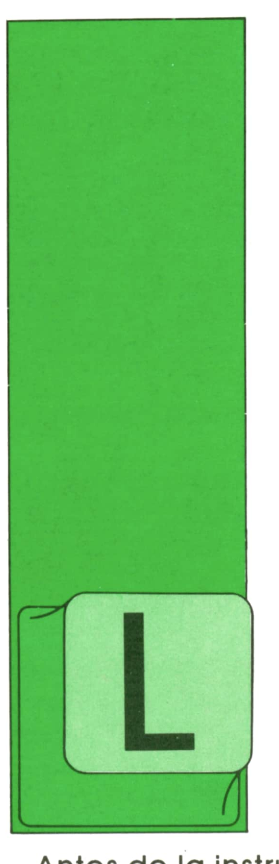




Fig. 7. Presentación en pantalla de una ficha en el programa 2.

MAQUINA 8088





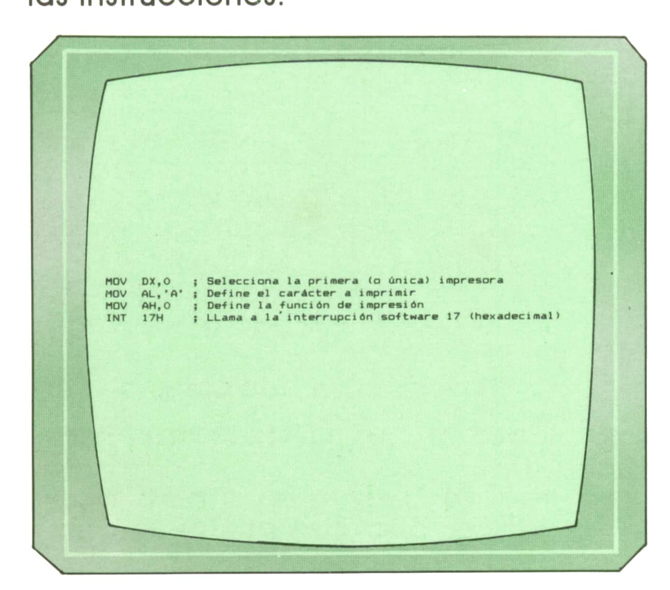
Manejo de la impresora (INT 17H)

La rutina del BIOS que está destinada a manejar la impresora (o las impresoras) esta asociada a la interrupción software número 23 (17 en hexadecimal).

Antes de la instrucción INT 17H hay que definir en el registro DX un 0, un 1 o un 2 para indicar cuál de las impresoras se quiere utilizar. En caso de tener una sola impresora debe definirse DX = 0.

Igualmente, antes de la instrucción INT 17H debe definirse en el registro AH un 0, un 1 o un 2 para indicar la función que se quiere realizar:

AH = 0. Imprime el carácter contenido en el registro AL. Ejemplo: si se desea imprimir el carácter "A", se pueden utilizar las instrucciones:



```
MOV  DX,0   ; Selecciona la primera (o única) impresora
MOV  AL,'A' ; Define el carácter a imprimir
MOV  AH,0   ; Define la función de impresión
INT  17H    ; LLama a la interrupción software 17 (hexadecimal)
```

Después de haberse ejecutado la instrucción INT 17H, el registro AH contiene el estado de la impresora. Si se ha devuelto AH = 1, debe interpretarse que la impresora no está preparada. Otros valores deben interpretarse igual que en la función de «Lectura del byte de estado» más abajo descrita.

AH = 1. Prepara la impresora para ser utilizada.

AH = 2. Lee el byte del estado de la impresora. Al terminar la instrucción INT 17H, el registro AH contiene el byte de estado. Los bits de este byte que estén en «on» (es decir, que valgan 1) deben interpretarse de la forma siguiente:

— El bit 0 indica que la impresora no responde.

— El bit 3 indica error de entrada/salida.

— El bit 5 indica que se ha acabado el papel.

— El bit 7 indica que la impresora está ocupada.



Manejo del teclado (INT 16H)

El BIOS maneja los caracteres generados por el teclado a través de dos rutinas asociadas a dos interrupciones diferentes.

La primera rutina está asociada a la interrupción hardware del teclado. Cada vez que se pulsa una tecla, se activa esta interrupción y la rutina se encarga de copiar la información de la tecla pulsada en un área de memoria del BIOS denominada «buffer del teclado».

La segunda rutina está asociada a la interrupción software número 22 (16 hexadecimal) y está destinada a ser lla-

mada por los programas en los que se utilice el teclado y contiene tres funciones diferentes.

La función a realizar se define (como en las demás interrupciones) en el registro AH.

AH = 0. Lee un carácter del teclado. Al terminar esta función, el registro AL contiene el carácter ASCII tecleado y AH contiene un código auxiliar denominado «scan code». En la mayoría de los casos el «scan code» coincide con el número de la tecla asociada al carácter. Las teclas de control no llevan asociadas ningún carácter ASCII, por esta razón devuelven AL = 0, lo cual indica que debe examinarse el «scan code» para distinguir cuál ha sido la tecla de control pulsada.

El caracter leído se elimina del buffer del teclado.

AH = 1. Comprueba si se ha pulsado alguna tecla y utiliza el flag ZF para indicarlo. En caso afirmativo, devuelve el citado flag en «off» y la tecla pulsada en AX (el código ASCII en AL y el «scan code» en AH). En caso negativo, devuelve el flag en «on». Esta función no elimina el carácter leído del buffer del teclado como hace la función anterior.

AH = 2. Devuelve en el registro AL el byte de estado del teclado. Los bits de dicho byte que están en «on» indican lo siguiente:

— El bit 0, que se ha pulsado la tecla «shift» del lado derecho.

— El bit 1, que se ha pulsado la tecla «shift» del lado izquierdo.

— El bit 2, que se han pulsado las teclas «Ctrl» y «Shift».

— El bit 3, que se han pulsado las teclas «Alt» y «Shift».

— El bit 4, que se ha pulsado la tecla «Scroll Lock».

— El bit 5, que se ha pulsado la tecla «Num Lock».

— El bit 6, que se ha pulsado la tecla «Caps Lock».

— El bit 7, que está activo el estado de «inserción de caracteres».

Manejo de las pantallas (INT 10H)

Las rutinas del BIOS que sirven para el manejo de las pantallas están asociadas a la interrupción 16 (10 hexadecimal) y contienen 16 funciones.

Antes de pasar a describir algunas de estas funciones es conveniente hacer algunas aclaraciones:

— Las funciones de manejo de caracteres de esta interrupción se pueden aplicar indistintamente a la pantalla monocroma y a la pantalla de color en cualquiera de sus modalidades.

— Las operaciones de lectura/escritura de caracteres de la pantalla se realizan en la posición que en ese momento tenga el cursor.

— La posición del cursor se define por medio de dos números que indican la fila y la columna contadas desde el vértice superior izquierdo, de modo que a dicho vértice le corresponde la fila 0, columna 0.

— Cada carácter en la pantalla lleva asociado un byte de atributo, que es un código con el que se pueden definir características como color del carácter, color de fondo, intensidad, parpadeo, etc.

— La pantalla de color en los modos alfabéticos permite tener definidas hasta 8 páginas de textos. Por esta razón, en las funciones 8, 9 y 10 hay que definir en el registro BH el número de la «página» sobre la que se quiere actuar, que debe ser un número comprendido entre 0 y 7. En los casos en que se utilice una sola página debe especificarse el BH = 0.

Igual que en las interrupciones anteriores, se utiliza el registro AH para definir la función deseada.

AH = 0. Establece la modalidad de funcionamiento de la pantalla. La modalidad deseada se especifica en el registro AL, de acuerdo con el siguiente convenio:

1. Pantalla de color. Modalidades alfabéticas.

— AL = 0 define el modo blanco y negro 40 × 25 (es decir, los caracteres se sitúan en 40 columnas y 25 filas).

— AL = 1 define el modo color 40 × 25.

— AL = 2 define el modo blanco y negro 80 × 25.

— AL = 3 define el modo color 80 × 25.

2. Pantalla de color. Modalidades gráficas.

— AL = 4 define el modo color 320 × 200 (es decir, los puntos se sitúan en 320 columnas y 200 filas).

— AL = 5 define el modo color 320 × 200.

— AL = 6 define el modo blanco y negro 640 × 200.

3. Pantalla monocroma.

— AL = 7 define el modo blanco y negro 80 × 25

AH = 2. Define la posición del cursor.

La fila debe especificarse en el registro DH y la columna en el registro DL.

AH = 3. Lee la posición del cursor.

La fila se obtiene en el registro DH y la columna en el registro DL.

AH = 8. Lee el atributo y el carácter contenidos a la posición donde esté situado el cursor.

Cuando se ejecuta esta función el registro AL contiene el código ASCII de dicho carácter y AH su atributo.

AH = 9. Escribe un carácter con un determinado atributo en la posición donde esté situado el cursor.

Antes de la llamada, el registro AL debe contener el código ASCII del carácter deseado y AH su atributo. CX debe contener el número de veces que se quiere repetir dicho carácter a partir de la posición del cursor.

AH = 10. Escribe un carácter sin modificar el atributo.

Igual que la función anterior, el registro AL debe contener el código ASCII del carácter deseado y CX debe contener el número de veces que se quiere repetir dicho carácter.

AH = 14. Escribe un carácter en modo teletipo.

Es decir, interpretando los códigos recibidos, y simulando en la pantalla el funcionamiento de un teletipo a efectos de salto de líneas, retorno de carro, paso atrás, etc.

AL debe contener el código ASCII del carácter deseado y BL el atributo.

AH = 15. Obtiene del BIOS datos acerca del modo de funcionamiento de la pantalla.

En el registro AL se obtiene el modo, que es un número de 0 a 7 que debe interpretarse con el mismo convenio que se explicó en la función AH = 0.



PROGRAMAS

EDUCATIVOS • DE UTILIDAD • DE GESTION • DE JUEGOS

Programa: Space Invader para Spectrum



POCO hay que comentar sobre este juego por todos conocido, salvo las teclas que tenemos que utilizar para mover nuestra nave y para disparar. Estas son:

Z — Izquierda
X — Derecha
M — Disparo

El programa se divide en dos partes. La primera nos permite introducir el código máquina en memoria y ejecutarlo. La segunda es el propio programa en código máquina y que es el juego en realidad.

Referente al primer programa, se advierte que, debido a la gran cantidad de líneas DATA, es muy fácil confundirse. Aunque el programa lleva un cheksum, hay que tener mucho cuidado a la hora de introducir el programa.

```
 10>REM ********************************
 20 REM *   S P A C E   I N V A D E R   *
 30 REM ********************************
 40 REM
 50 REM ********************************
 60 REM * (c) Ediciones Siglo Cultural *
 70 REM * (c) 1987                     *
 80 REM ********************************
 90 REM
100 BORDER 0
110 PAPER 0
120 INK 6
130 CLS
140 CLEAR 54999
145 LET TOT=0
160 RESTORE 1000
170 PRINT AT 10,8; FLASH 1;"ESPERE UN MOMENTO"
180 PRINT AT 12,9; FLASH 1; INVERSE 1; INK 4;"CARGANDO DATAS."
190 FOR I=55000 TO 56906
200    READ A
205    LET TOT=TOT+A
210    POKE I,A
220 NEXT I
```

```
225 IF TOT<>215743 THEN  GO TO 500
230 PRINT AT 17,2; INK 5;"PULSE UNA TECLA PARA EMPEZAR"
235 PAUSE 0
240 CLS
250 PRINT AT 0,0; PAPER 7; INK 1;"PUNTOS";
260 PRINT " 0"
270 PRINT AT 0,25; PAPER 7; INK 1;"VIDAS";
280 PRINT " 3"
300 RANDOMIZE USR 55000
310 PRINT AT 0,31;"0"
320 PRINT AT 11,12; FLASH 1;"GAME OVER"
330 FOR I=1 TO 400: PAUSE 1: NEXT I
340 GO TO 230
500 REM
510 REM ************************
520 REM * ERROR EN LINEAS DATA *
530 REM ************************
540 REM
550 CLS
560 PRINT "LAS LINEAS DATAS NO ESTAN CORRECTAMENTE ESCRITAS"
570 PRINT
580 PRINT "REPASALAS ANTES DE CONTINUAR."
590 GO TO 9999
990 REM
991 REM ***************
992 REM * LINEAS DATA *
993 REM ***************
994 REM
1000 DATA 62,2,205,1,22,175,50,80,222,50,81,222
1005 DATA 50,93,222,62,3,50,100,222,62,37,50,97
1010 DATA 222,33,0,241,17,1,241,54,255,1,254,0
1015 DATA 237,176,175,50,82,222,50,77,222,50,79,222
1020 DATA 205,183,217,205,183,220,205,55,220,62,7,8
1025 DATA 17,3,1,237,83,90,222,17,0,23,33,116
1030 DATA 221,205,110,220,175,50,78,222,50,85,222,58
1035 DATA 97,222,50,96,222,62,5,50,99,222,175,50
1040 DATA 98,222,62,127,219,254,230,1,200,118,205,210
1045 DATA 216,33,98,222,52,62,1,166,40,6,205,241
1050 DATA 218,205,1,216,33,96,222,53,32,15,58,97
1055 DATA 222,50,96,222,205,99,219,58,89,222,254,22
1060 DATA 200,33,99,222,53,32,8,54,5,205,25,218
1065 DATA 205,37,217,58,78,222,61,202,202,215,58,92
1070 DATA 222,167,32,182,33,93,222,126,60,254,7,32
1075 DATA 1,175,119,17,1,72,33,0,72,1,255,15
1080 DATA 237,176,58,77,222,167,202,236,214,1,2,1
1085 DATA 237,67,90,222,33,188,221,58,76,222,95,22
1090 DATA 1,205,110,220,195,236,214,62,22,215,175,215
1095 DATA 62,7,215,237,75,80,222,205,43,45,205,227
1100 DATA 45,201,1,0,0,237,95,211,254,11,120,177
1105 DATA 32,247,175,211,254,33,100,222,53,200,62,22
1110 DATA 215,62,0,215,62,31,215,126,198,48,215,1
1115 DATA 3,1,33,188,221,237,67,90,222,22,23,58
1120 DATA 85,222,95,205,110,220,195,17,215,62,3,8
1125 DATA 58,77,222,167,40,51,58,76,222,95,22,1
1130 DATA 33,188,221,213,1,2,1,237,67,90,222,205
1135 DATA 110,220,209,28,123,254,31,40,19,50,76,222
1140 DATA 33,60,222,205,110,220,33,171,9,17,1,0
1145 DATA 205,181,3,201,175,50,77,222,201,237,95,230
1150 DATA 255,192,62,1,50,77,222,61,50,76,222,17
1155 DATA 0,1,1,2,1,237,67,90,222,33,60,222
1160 DATA 205,110,220,201,58,77,222,167,200,58,76,222
1165 DATA 79,187,40,3,60,187,192,89,33,188,221,17
1170 DATA 2,1,237,83,90,222,205,110,220,237,95,230
1175 DATA 3,33,144,216,6,0,79,9,78,42,80,222
1180 DATA 9,34,80,222,205,183,215,201,10,50,150,200
1185 DATA 17,1,242,33,0,242,1,127,0,54,0,237
1190 DATA 176,17,4,1,237,83,90,222,17,2,19,33
1195 DATA 188,221,6,4,197,213,6,4,197,213,229,205
1200 DATA 110,220,225,209,20,193,16,244,209,193,62,8
```

```
1205 DATA 131,95,16,232,33,156,1,17,19,0,205,181
1210 DATA 3,201,58,82,222,167,200,33,1,1,34,90
1215 DATA 222,62,7,8,58,84,222,95,58,83,222,87
1220 DATA 205,141,217,56,5,175,50,82,222,201,33,188
1225 DATA 221,213,205,110,220,209,21,122,254,1,32,5
1230 DATA 205,92,216,24,232,50,83,222,205,141,217,48
1235 DATA 224,213,205,120,218,209,58,82,222,167,200,33
1240 DATA 180,221,1,1,1,237,67,90,222,205,110,220
1245 DATA 201,58,79,222,167,200,17,1,1,237,83,90
1250 DATA 222,71,33,0,241,62,7,8,126,44,94,44
1255 DATA 254,255,40,248,197,45,45,87,229,213,33,188
1260 DATA 221,205,110,220,209,225,20,122,254,23,32,14
1265 DATA 205,120,217,54,255,58,79,222,61,50,79,222
1270 DATA 24,16,205,141,217,48,240,114,229,213,33,180
1275 DATA 221,205,110,220,209,225,44,44,193,16,193,201
1280 DATA 58,85,222,187,40,9,60,187,40,5,60,187
1285 DATA 40,1,201,62,1,50,78,222,201,122,214,19
1290 DATA 216,7,7,7,7,7,131,79,6,242,10,167
1295 DATA 40,23,175,2,229,98,107,205,146,220,6,8
1300 DATA 237,95,230,231,166,119,36,16,247,55,63,225
1305 DATA 201,55,201,33,0,242,17,1,242,1,127,0
1310 DATA 54,0,237,176,6,16,33,2,242,84,93,28
1315 DATA 197,1,3,0,54,1,237,176,193,125,198,5
1320 DATA 111,16,238,62,5,8,17,4,1,237,83,90
1325 DATA 222,6,4,17,2,19,205,244,217,62,8,131
1330 DATA 95,16,247,201,197,213,33,220,221,213,205,110
1335 DATA 220,209,20,33,252,221,213,229,205,110,220,225
1340 DATA 209,20,213,205,110,220,209,33,28,222,20,205
1345 DATA 110,220,209,193,201,58,93,222,60,7,71,58
1350 DATA 79,222,184,200,58,89,222,254,21,200,237,95
1355 DATA 230,7,79,237,95,230,1,129,79,129,129,38
1360 DATA 240,111,6,4,86,44,94,44,126,230,128,32
1365 DATA 7,17,25,0,25,16,241,201,33,0,241,126
1370 DATA 254,255,40,4,44,44,24,247,20,20,114,237
1375 DATA 95,230,1,131,44,119,95,33,180,221,62,7
1380 DATA 8,1,1,1,237,67,90,222,205,110,220,33
1385 DATA 79,222,52,201,58,82,222,167,200,17,2,2
1390 DATA 237,83,90,222,33,0,240,58,92,222,71,58
1395 DATA 84,222,95,58,83,222,87,197,70,44,78,44
1400 DATA 126,44,230,128,40,246,120,186,40,4,60,186
1405 DATA 32,71,121,187,40,4,60,187,32,63,45,54
1410 DATA 0,33,148,221,80,89,213,205,110,220,33,106
1415 DATA 6,17,5,0,205,181,3,209,1,0,160,11
1420 DATA 120,177,32,251,33,188,221,205,110,220,42,80
1425 DATA 222,1,10,0,9,34,80,222,205,183,215,33
1430 DATA 92,222,53,175,50,82,222,193,33,97,222,53
1435 DATA 201,193,16,163,201,17,3,1,237,83,90,222
1440 DATA 22,23,58,85,222,95,62,7,8,33,188,221
1445 DATA 219,223,254,1,40,41,254,2,40,28,254,16
1450 DATA 40,58,62,127,219,254,230,4,40,50,14,254
1455 DATA 65,237,64,62,2,160,40,6,62,4,160,40
1460 DATA 10,201,123,167,200,205,70,219,29,24,8,123
1465 DATA 254,29,200,205,70,219,28,33,116,221,123,50
1470 DATA 85,222,205,110,220,201,213,205,110,220,209,201
1475 DATA 58,82,222,61,200,58,85,222,60,50,84,222
1480 DATA 62,22,50,83,222,62,1,50,82,222,201,58
1485 DATA 89,222,254,17,32,7,58,94,222,61,204,148
1490 DATA 216,205,174,219,33,242,12,17,1,0,205,181
1495 DATA 3,58,94,222,254,1,40,29,58,87,222,254
1500 DATA 30,40,7,58,88,222,167,40,1,201,58,95
1505 DATA 222,50,86,222,175,50,95,222,60,50,94,222
1510 DATA 201,58,86,222,237,68,50,95,222,175,50,94
1515 DATA 222,201,175,50,87,222,50,89,222,61,50,88
1520 DATA 222,17,2,2,237,83,90,222,33,0,240,58
1525 DATA 94,222,87,58,95,222,95,6,4,62,2,8
1530 DATA 197,6,9,197,213,213,229,86,44,94,44,126
1535 DATA 44,230,128,40,82,33,188,221,213,205,110,220
```

```
1540 DATA 193,225,209,120,130,119,87,44,121,131,119,95
1545 DATA 44,126,79,58,87,222,187,48,4,123,50,87
1550 DATA 222,58,88,222,187,56,4,123,50,88,222,58
1555 DATA 89,222,186,48,4,122,50,89,222,121,238,32
1560 DATA 119,44,229,230,127,79,6,0,33,244,220,9
1565 DATA 205,110,220,225,209,193,16,167,193,8,60,8
1570 DATA 16,158,201,193,209,24,241,17,2,2,237,83
1575 DATA 90,222,6,4,62,2,8,33,0,240,197,6
1580 DATA 9,197,86,44,94,44,126,44,71,229,230,128
1585 DATA 40,13,33,244,220,120,230,127,6,0,79,9
1590 DATA 205,110,220,225,193,16,226,193,8,60,8,16
1595 DATA 217,201,235,237,75,90,222,197,229,205,146,220
1600 DATA 6,8,229,26,119,36,19,16,250,225,44,217
1605 DATA 8,18,28,8,217,13,32,236,225,193,36,16
1610 DATA 226,201,197,68,77,120,230,24,246,64,103,120
1615 DATA 230,7,7,7,7,7,7,129,111,193,229,217
1620 DATA 209,122,230,24,203,47,203,47,203,47,246,88
1625 DATA 87,217,201,58,93,222,198,11,33,0,240,95
1630 DATA 14,224,22,0,6,4,197,213,6,9,115,44
1635 DATA 114,44,121,238,32,79,119,44,20,20,20,16
1640 DATA 241,209,193,123,214,3,95,121,238,64,79,16
1645 DATA 225,62,36,50,92,222,62,1,50,95,222,175
1650 DATA 50,94,222,201,8,4,2,7,15,25,49,121
1655 DATA 16,32,64,224,240,152,140,158,127,63,48,31
1660 DATA 31,32,16,8,254,252,12,248,248,4,8,16
1665 DATA 2,4,2,7,15,25,49,121,64,32,64,224
1670 DATA 240,152,140,158,127,59,60,31,31,32,64,128
1675 DATA 254,220,60,248,248,4,2,1,0,15,31,63
1680 DATA 127,99,127,127,0,240,248,252,254,198,254,254
1685 DATA 127,115,50,24,12,6,0,0,254,206,76,24
1690 DATA 48,96,0,0,0,15,31,63,127,99,127,127
1695 DATA 0,240,248,252,254,198,254,254,127,115,50,16
1700 DATA 24,8,8,0,254,206,76,8,24,16,16,0
1705 DATA 0,0,1,7,63,127,112,96,24,60,255,255
1710 DATA 255,255,0,0,0,0,128,224,252,254,14,6
1715 DATA 0,120,68,66,66,68,120,0,0,59,106,85
1720 DATA 42,53,26,13,0,16,168,88,176,88,168,84
1725 DATA 26,53,87,45,56,16,0,0,172,84,234,86
1730 DATA 136,4,0,0,24,24,24,24,24,24,24,24
1735 DATA 0,0,0,0,0,0,0,0,0,0,0,0
1740 DATA 0,0,0,0,0,0,0,0,0,0,0,0
1745 DATA 0,0,0,0,0,0,0,0,1,3,7,15
1750 DATA 31,63,127,255,255,255,255,255,255,255,255,255
1755 DATA 255,255,255,255,255,255,255,255,128,192,224,240
1760 DATA 248,252,254,255,255,255,255,255,255,255,255,255
1765 DATA 255,255,255,255,255,255,255,255,255,255,255,255
1770 DATA 255,255,255,255,255,255,255,255,255,255,255,255
1775 DATA 255,254,252,248,240,224,192,192,0,0,0,0
1780 DATA 0,0,0,0,0,0,0,0,0,0,0,0
1785 DATA 255,127,63,31,15,7,3,3,1,3,11,27
1790 DATA 12,23,35,96,128,192,208,152,48,232,196,0
1795 DATA 0,0,0,0,0,0,0,0,0,0,0,0
```

Para todos aquellos que entienden de código máquina, se incluye el listado fuente del programa.

```
1000 ;***************
1001 ;*             *
1002 ;*  S P A C E  *
1003 ;*   INVADER   *
1004 ;*             *
1005 ;***************
1006 ;
1007        ORG  55000
1008        LD   A,2
1009        CALL #1601
1010        XOR  A
1011        LD   (SCORE),A
1012        LD   (SCORE+1),A
1013        LD   (LEVEL),A
1014        LD   A,3
1015        LD   (VIDAS),A
1016 NEWLEV LD   A,37
1017        LD   (CON1),A
1018        LD   HL,#F100
1019        LD   DE,#F101
1020        LD   (HL),#FF
1021        LD   BC,254
1022        LDIR
1023        XOR  A
1024        LD   (FLGD),A
1025        LD   (OVNF),A
1026        LD   (NDISP),A
1027        CALL SETUPB
1028        CALL CREAT'
1029        CALL SACAM
1030 SETVID LD   A,7
1031        EX   AF,AF
1032        LD   DE,#0103
1033        LD   (TAM),DE
1034        LD   DE,#1700
1035        LD   HL,GRAFB
1036        CALL SPRITE
1037        XOR  A
1038        LD   (MFLAG),A
1039        LD   (CBX),A
1040        LD   A,(CON1)
1041        LD   (CON),A
1042        LD   A,5
1043        LD   (CICL2),A
1044        XOR  A
1045        LD   (CICLOS),A
1046 PLAY
1047        LD   A,#7F
1048        IN   A,(254)
1049        AND  1
1050        RET  Z
1051        HALT
1052        CALL MOVDB
1053        LD   HL,CICLOS
1054        INC  (HL)
1055        LD   A,1
1056        AND  (HL)
1057        JR   Z,PLAY0
1058        CALL MUEVEB
1059        CALL MOVOV
1060 PLAY0  LD   HL,CON
1061        DEC  (HL)
1062        JR   NZ,PLAY2
1063        LD   A,(CON1)
1064        LD   (CON),A
1065        CALL BLOQUE
1066        LD   A,(MAYY)
1067        CP   22
1068        RET  Z
1069 PLAY2  LD   HL,CICL2
1070        DEC  (HL)
1071        JR   NZ,PLAY3
1072        LD   (HL),5
1073        CALL DISPAL
1074        CALL MUEVD
1075 PLAY3
1076        LD   A,(MFLAG)
1077        DEC  A
1078        JP   Z,VIDOUT
1079        LD   A,(NMAR)
1080        AND  A
1081        JR   NZ,PLAY
1082        LD   HL,LEVEL
1083        LD   A,(HL)
1084        INC  A
1085        CP   7
1086        JR   NZ,PLAY5
1087        XOR  A
1088 PLAY5  LD   (HL),A
1089        LD   DE,18433
1090        LD   HL,18432
1091        LD   BC,4095
1092        LDIR
1093        LD   A,(OVNF)
1094        AND  A
1095        JP   Z,NEWLEV
1096        LD   BC,#0102
1097        LD   (TAM),BC
1098        LD   HL,VACIO
1099        LD   A,(POSOV)
1100        LD   E,A
1101        LD   D,1
1102        CALL SPRITE
1103        JP   NEWLEV
1104 PRTSC  LD   A,22
1105        RST  16
1106        XOR  A
1107        RST  16
1108        LD   A,7
1109        RST  16
1110        LD   BC,(SCORE)
1111        CALL #2D2B
1112        CALL #2DE3
1113        RET
1114 VIDOUT LD   BC,0
1115 RUIDO  LD   A,R
1116        OUT  (254),A
1117        DEC  BC
1118        LD   A,B
1119        OR   C
1120        JR   NZ,RUIDO
1121        XOR  A
1122        OUT  (254),A
1123        LD   HL,VIDAS
1124        DEC  (HL)
1125        RET  Z
1126        LD   A,22
```

```
        RST  16
        LD   A,0
        RST  16
        LD   A,31
        RST  16
        LD   A,(HL)
        ADD  A,48
        RST  16
        LD   BC,#0103
        LD   HL,VACIO
        LD   (TAM),BC
        LD   D,23
        LD   A,(CBX)
        LD   E,A
        CALL SPRITE
        JP   SETVID
MOVOV
        LD   A,3
        EX   AF,AF
        LD   A,(OVNF)
        AND  A
        JR   Z,CREAOV
        LD   A,(POSOV)
        LD   E,A
        LD   D,1
        LD   HL,VACIO
        PUSH DE
        LD   BC,#0102
        LD   (TAM),BC
        CALL SPRITE
        POP  DE
        INC  E
        LD   A,E
        CP   31
        JR   Z,DESAP
        LD   (POSOV),A
        LD   HL,OVNI
        CALL SPRITE
        LD   HL,OVS1
        LD   DE,OVS2
        CALL SOUND
        RET
DESAP   XOR  A
        LD   (OVNF),A
        RET
CREAOV  LD   A,R
        AND  #FF
        RET  NZ
        LD   A,1
        LD   (OVNF),A
        DEC  A
        LD   (POSOV),A
        LD   DE,#0100
        LD   BC,#0102
        LD   (TAM),BC
        LD   HL,OVNI
        CALL SPRITE
        RET
OVS1    EQU  2475
OVS2    EQU  1
OVNTES
        LD   A,(OVNF)
        AND  A
        RET  Z
        LD   A,(POSOV)
        LD   C,A
        CP   E
        JR   Z,OVS
        INC  A
        CP   E
        RET  NZ
OVS     LD   E,C
        LD   HL,VACIO
        LD   DE,#0102
        LD   (TAM),DE
        CALL SPRITE
        LD   A,R
        AND  3
        LD   HL,TABPU
        LD   B,0
        LD   C,A
        ADD  HL,BC
        LD   C,(HL)
        LD   HL,(SCORE)
        ADD  HL,BC
        LD   (SCORE),HL
        CALL PRTSC
        RET
TABPU   DEFB 10,50,150,200
SUPRB
        LD   DE,#F201
        LD   HL,#F200
        LD   BC,127
        LD   (HL),0
        LDIR
;SUPRIME BARRICADAS
        LD   DE,#0104
        LD   (TAM),DE
        LD   DE,#1302
        LD   HL,VACIO
        LD   B,4
BUCAN   PUSH BC
        PUSH DE
        LD   B,4
BUCAN1  PUSH BC
        PUSH DE
        PUSH HL
        CALL SPRITE
        POP  HL
        POP  DE
        INC  D
        POP  BC
        DJNZ BUCAN1
        POP  DE
        POP  BC
        LD   A,8
        ADD  A,E
        LD   E,A
        DJNZ BUCAN
        LD   HL,SON1
        LD   DE,SON2
        CALL SOUND
        RET
SON1    EQU  412
SON2    EQU  19
;
; MOVIMIENTO DISPARO BASE
;
MOVDB   LD   A,(FLGD)
        AND  A
        RET  Z
        LD   HL,#0101
        LD   (TAM),HL
        LD   A,7
```

```
                AF,AF
                A,(PXD)
          LD    E,A
          LD    A,(PYD)
          LD    D,A
          CALL  COLIB
          JR    C,SIG4
SIG3      XOR   A
          LD    (FLGD),A
          RET
SIG4      LD    HL,VACIO
          PUSH  DE
          CALL  SPRITE
          POP   DE
          DEC   D
          LD    A,D
          CP    1
          JR    NZ,SIG5
          CALL  OVNTES
          JR    SIG3
SIG5      LD    (PYD),A
          CALL  COLIB
          JR    NC,SIG3
          PUSH  DE
          CALL  DETEC
          POP   DE
          LD    A,(FLGD)
          AND   A
          RET   Z
          LD    HL,TIRO
          LD    BC,#0101
          LD    (TAM),BC
          CALL  SPRITE
          RET
;
;MOVI.DISP ALIENS
;
; EL FLAG MFLAG=1
;SI HA HABIDO UN
; CRIMEN
;
MUEVD     LD    A,(NDISP)
          AND   A
          RET   Z
          LD    DE,#0101
          LD    (TAM),DE
          LD    B,A
          LD    HL,#F100
          LD    A,7
          EX    AF,AF
MISIL     LD    A,(HL)
          INC   L
          LD    E,(HL)
          INC   L
          CP    #FF
          JR    Z,MISIL
          PUSH  BC
          DEC   L
          DEC   L
          LD    D,A
          PUSH  HL
          PUSH  DE
          LD    HL,VACIO
          CALL  SPRITE
          POP   DE
          POP   HL
          INC   D
          LD    A,D
          CP    23
          JR    NZ,NOSUEL
          CALL  NAVE
KAPUT     LD    (HL),#FF
          LD    A,(NDISP)
          DEC   A
          LD    (NDISP),A
          JR    MISIL3
NOSUEL    CALL  COLIB
          JR    NC,KAPUT
          LD    (HL),D
          PUSH  HL
          PUSH  DE
          LD    HL,TIRO
          CALL  SPRITE
          POP   DE
          POP   HL
MISIL3    INC   L
          INC   L
          POP   BC
          DJNZ  MISIL
          RET
NAVE      LD    A,(CBX)
          CP    E
          JR    Z,MUERTE
          INC   A
          CP    E
          JR    Z,MUERTE
          INC   A
          CP    E
          JR    Z,MUERTE
          RET
MUERTE    LD    A,1
          LD    (MFLAG),A
          RET
COLIB     LD    A,D
          SUB   19
          RET   C
          RLCA
          RLCA
          RLCA
          RLCA
          RLCA
          ADD   A,E
          LD    C,A
          LD    B,#F2
          LD    A,(BC)
          AND   A
          JR    Z,SALIR
          XOR   A
          LD    (BC),A
          PUSH  HL
          LD    H,D
          LD    L,E
          CALL  GETDIR
          LD    B,8
BUS8
          LD    A,R
          AND   %11100111
          AND   (HL)
          LD    (HL),A
          INC   H
          DJNZ  BUS8
          SCF
          CCF
          POP   HL
```

```
        RET
SALIR   SCF
        RET
SETUPB  LD    HL,#F200
        LD    DE,#F201
        LD    BC,127
        LD    (HL),0
        LDIR
        LD    B,16
        LD    HL,#F202
SETB    LD    D,H
        LD    E,L
        INC   E
        PUSH  BC
        LD    BC,3
        LD    (HL),1
        LDIR
        POP   BC
        LD    A,L
        ADD   A,5
        LD    L,A
        DJNZ  SETB
        LD    A,5
        EX    AF,AF
        LD    DE,#0104
        LD    (TAM),DE
        LD    B,4
        LD    DE,#1302
SET2    CALL  PRINTB
        LD    A,8
        ADD   A,E
        LD    E,A
        DJNZ  SET2
        RET
PRINTB  PUSH  BC
        PUSH  DE
        LD    HL,B1
        PUSH  DE
        CALL  SPRITE
        POP   DE
        INC   D
        LD    HL,B2
        PUSH  DE
        PUSH  HL
        CALL  SPRITE
        POP   HL
        POP   DE
        INC   D
        PUSH  DE
        CALL  SPRITE
        POP   DE
        LD    HL,B3
        INC   D
        CALL  SPRITE
        POP   DE
        POP   BC
        RET
;
; DISPARO DE ALIENS
;
DISPAL  LD    A,(LEVEL)
        INC   A
        RLCA
        LD    B,A
        LD    A,(NDISP)
        CP    B
        RET   Z
        LD    A,(MAYY)
        CP    21
        RET   Z
        LD    A,R
        AND   7
        LD    C,A
        LD    A,R
        AND   1
        ADD   A,C
        LD    C,A
        ADD   A,C
        ADD   A,C
        LD    H,#F0
        LD    L,A
        LD    B,4
BUSC    LD    D,(HL)
        INC   L
        LD    E,(HL)
        INC   L
        LD    A,(HL)
        AND   128
        JR    NZ,ENC
        LD    DE,25
        ADD   HL,DE
        DJNZ  BUSC
        RET
ENC     LD    HL,#F100
PAZ     LD    A,(HL)
        CP    #FF
        JR    Z,ENC2
        INC   L
        INC   L
        JR    PAZ
ENC2    INC   D
        INC   D
        LD    (HL),D
        LD    A,R
        AND   1
        ADD   A,E
        INC   L
        LD    (HL),A
        LD    E,A
        LD    HL,TIRO
        LD    A,7
        EX    AF,AF
        LD    BC,#0101
        LD    (TAM),BC
        CALL  SPRITE
        LD    HL,NDISP
        INC   (HL)
        RET
TABDIS  EQU   #F100
;
; DETECCION DE
; COLISIONES
;
DETEC   LD    A,(FLGD)
        AND   A
        RET   Z
        LD    DE,#0202
        LD    (TAM),DE
        LD    HL,TABM
        LD    A,(NMAR)
        LD    B,A
        LD    A,(PXD)
        LD    E,A
        LD    A,(PYD)
```

```
        LD    D,A
COLIS   PUSH  BC
COLIS1  LD    B,(HL)
        INC   L
        LD    C,(HL)
        INC   L
        LD    A,(HL)
        INC   L
        AND   128
        JR    Z,COLIS1
        LD    A,B
        CP    D
        JR    Z,SIT1
        INC   A
        CP    D
        JR    NZ,NOCOM
SIT1    LD    A,C
        CP    E
        JR    Z,SIT
        INC   A
        CP    E
        JR    NZ,NOCOM
SIT
        DEC   L
        LD    (HL),0
        LD    HL,EXPLOS
        LD    D,B
        LD    E,C
        PUSH  DE
        CALL  SPRITE
        LD    HL,ESP1
        LD    DE,ESP2
        CALL  SOUND
        POP   DE
        LD    BC,#A000
ESP     DEC   BC
        LD    A,B
        OR    C
        JR    NZ,ESP
        LD    HL,VACIO
        CALL  SPRITE
        LD    HL,(SCORE)
        LD    BC,10
        ADD   HL,BC
        LD    (SCORE),HL
        CALL  PRTSC
        LD    HL,NMAR
        DEC   (HL)
        XOR   A
        LD    (FLGD),A
        POP   BC
        LD    HL,CON1
        DEC   (HL)
        RET
NOCOM
        POP   BC
        DJNZ  COLIS
        RET
ESP1    EQU   1642
ESP2    EQU   5
SOUND   EQU   #03B5
MUEVEB  LD    DE,#0103
        LD    (TAM),DE
        LD    D,23
        LD    A,(CBX)
        LD    E,A
        LD    A,7
        EX    AF,AF
        LD    HL,VACIO
        IN    A,(223)
        CP    1
        JR    Z,DER
        CP    2
        JR    Z,IZQ
        CP    16
        JR    Z,DISP
        LD    A,#7F
        IN    A,(254)
        AND   4
        JR    Z,DISP
        LD    C,254
        LD    B,C
        IN    B,(C)
        LD    A,2
        AND   B
        JR    Z,IZQ
        LD    A,4
        AND   B
        JR    Z,DER
        RET
IZQ     LD    A,E
        AND   A
        RET   Z
        CALL  BORRAN
        DEC   E
        JR    CONT
DER     LD    A,E
        CP    29
        RET   Z
        CALL  BORRAN
        INC   E
CONT    LD    HL,GRAFB
        LD    A,E
        LD    (CBX),A
        CALL  SPRITE
        RET
BORRAN  PUSH  DE
        CALL  SPRITE
        POP   DE
        RET
DISP
        LD    A,(FLGD)
        DEC   A
        RET   Z
        LD    A,(CBX)
        INC   A
        LD    (PXD),A
        LD    A,22
        LD    (PYD),A
        LD    A,1
        LD    (FLGD),A
        RET
BLOQUE
        LD    A,(MAYY)
        CP    17
        JR    NZ,CONTI
        LD    A,(INCY)
        DEC   A
        CALL  Z,SUPRB
CONTI   CALL  MUEVEM
        LD    HL,3314
        LD    DE,1
        CALL  SOUND
        LD    A,(INCY)
```

```
1663        CP    1
1664        JR    Z,NEWSET
1665        LD    A,(MAYX)
1666        CP    30
1667        JR    Z,BAJA
1668        LD    A,(MENX)
1669        AND   A
1670        JR    Z,BAJA
1671        RET
1672 BAJA   LD    A,(INCX)
1673        LD    (OLDS),A
1674        XOR   A
1675        LD    (INCX),A
1676        INÇ   A
1677        LD    (INCY),A
1678        RET
1679 NEWSET
1680        LD    A,(OLDS).
1681        NEG
1682        LD    (INCX),A
1683        XOR   A
1684        LD    (INCY),A
1685        RET
1686 MUEVEM
1687        XOR   A
1688        LD    (MAYX),A
1689        LD    (MAYY),A
1690        DEC   A
1691        LD    (MENX),A
1692        LD    DE,#0202
1693        LD    (TAM),DE
1694        LD    HL,TABM
1695        LD    A,(INCY)
1696        LD    D,A
1697        LD    A,(INCX)
1698        LD    E,A
1699        LD    B,4
1700        LD    A,2
1701        EX    AF,AF
1702 BCL1   PUSH  BC
.1703       LD    B,9
1704 BCL2   PUSH  BC
1705        PUSH  DE
1706        PUSH  DE
1707        PUSH  HL
1708        LD    D,(HL)
1709        INC   L
1710        LD    E,(HL)
1711        INC   L
1712        LD    A,(HL)
1713        INC   L
1714        AND   128
1715        JR    Z,MUERT
1716        LD    HL,VACIO
1717        PUSH  DE
1718        CALL  SPRITE
1719        POP   BC
1720        POP   HL
1721        POP   DE
1722        LD    A,B
1723        ADD   A,D
1724        LD    (HL),A
1725        LD    D,A
1726        INC   L
1727        LD    A,C
1728        ADD   A,E
1729        LD    (HL),A
1730        LD    E,A
1731        INC   L
1732        LD    A,(HL)
1733        LD    C,A
1734        LD    A,(MAYX)
1735        CP    E
1736        JR    NC,NMAY
1737        LD    A,E
1738        LD    (MAYX),A
1739 NMAY   LD    A,(MENX)
1740        CP    E
1741        JR    C,NOMEN
1742        LD    A,E
1743        LD    (MENX),A
1744 NOMEN  LD    A,(MAYY)
1745        CP    D
1746        JR    NC,NOMAYY
1747        LD    A,D
1748        LD    (MAYY),A
1749 NOMAYY LD    A,C
1750        XOR   32
1751        LD    (HL),A
1752        INC   L
1753        PUSH  HL
1754        AND   127
1755        LD    C,A
1756        LD    B,0
1757        LD    HL,GRAFM
1758        ADD   HL,BC
1759 CONT2  CALL  SPRITE
1760        POP   HL
1761 MUERT2
1762        POP   DE
1763        POP   BC
1764        DJNZ  BCL2
1765        POP   BC
1766        EX    AF,AF
1767        INC   A
1768        EX    AF,AF
1769        DJNZ  BCL1
1770        RET
1771 MUERT  POP   BC
1772        POP   DE
1773        JR    MUERT2
C1774 SACAM
1775        LD    DE,#0202
1776        LD    (TAM),DE
1777        LD    B,4
1778        LD    A,2
1779        EX    AF,AF
1780        LD    HL,TABM
1781 BUCLE1 PUSH  BC
1782        LD    B,9
1783 BUCLE
1784        PUSH  BC
1785        LD    D,(HL)
1786        INC   L
1787        LD    E,(HL)
1788        INC   L
1789        LD    A,(HL)
1790        INC   L
1791        LD    B,A
1792        PUSH  HL
1793        AND   128
1794        JR    Z,RIP
1795        LD    HL,GRAFM
1796        LD    A,B
```

```
        AND  127
        LD   B,0
        LD   C,A
        ADD  HL,BC
        CALL SPRITE
RIP     POP  HL
        POP  BC
        DJNZ BUCLE
        POP  BC
        EX   AF,AF
        INC  A
        EX   AF,AF
        DJNZ BUCLE1
        RET
;
;RUTINA DE SPRITES
; HL=DIR.GRAFICO
; DE=COORDS
; D=CORRDY
; E=COORDX
;
SPRITE
        EX   DE,HL
        LD   BC,(TAM) ; B=T
PON2    PUSH BC
        PUSH HL
        CALL GETDIR
PON4    LD   B,8
        PUSH HL
PON3    LD   A,(DE)
        LD   (HL),A
        INC  H
        INC  DE
        DJNZ PON3
        POP  HL
        INC  L
        EXX
        EX   AF,AF
        LD   (DE),A
        INC  E
        EX   AF,AF
        EXX
        DEC  C
        JR   NZ,PON4
        POP  HL
        POP  BC
        INC  H
        DJNZ PON2
        RET
GETDIR
        PUSH BC
        LD   B,H
        LD   C,L
        LD   A,B
        AND  24
        OR   64
        LD   H,A
        LD   A,B
        AND  7
        RLCA
        RLCA
        RLCA
        RLCA
        RLCA
        ADD  A,C
        LD   L,A
        POP  BC
```

```
        PUSH HL
        EXX
        POP  DE
        LD   A,D
        AND  #18
        SRA  A
        SRA  A
        SRA  A
        OR   #58
        LD   D,A
        EXX
        RET
;
; CREA TABLA MARCIANOS
;
CREAT
        LD   A,(LEVEL)
        ADD  A,11
        LD   HL,TABM
        LD   E,A
        LD   C,%11100000
        LD   D,0
        LD   B,4
BUC2
        PUSH BC
        PUSH DE
        LD   B,9
BUC1
        LD   (HL),E
        INC  L
        LD   (HL),D
        INC  L
             A,C
        XOR  32
        LD   C,A
        LD   (HL),A
        INC  L
        INC  D
        INC  D
        INC  D
        DJNZ BUC1
        POP  DE
        POP  BC
        LD   A,E
        SUB  3
        LD   E,A
        LD   A,C
        XOR  64
        LD   C,A
        DJNZ BUC2
        LD   A,36
        LD   (NMAR),A
        LD   A,1
        LD   (INCX),A
        XOR  A
        LD   (INCY),A
        RET
TABM    EQU  #F000
GRAFM
        DEFB 8,4,2,7
        DEFB 15,25,49,121
        DEFB 16,32,64,224
        DEFB 240,152,140,15
        DEFB 127,63,48,31
        DEFB 31,32,16,8
        DEFB 254,252,12,248
        DEFB 248,4,8,16
```

```
1931 ;
1932 ; SEGUNDO GRAFICO
1933 ;
1934        DEFB 2,4,2,7
1935        DEFB 15,25,49,121
1936        DEFB 64,32,64,224
1937        DEFB 240,152,140,15        8
1938        DEFB 127,59,60,31
1939        DEFB 31,32,64,128
1940        DEFB 254,220,60,248
1941        DEFB 248,4,2,1
1942 ;
1943 ; SEGUNDO MARCIANO
1944 ;
1945 SEGMAR DEFB 0,15,31,63
1946        DEFB 127,99,127,127
1947        DEFB 0,240,248,252
            DEFB 254,198,254,25        4
            DEFB 127,115,50,24
1950        DEFB 12,6,0,0
1951        DEFB 254,206,76,24
1952        DEFB 48,96,0,0
1953 ;
1954 ; SEGUNDA FASE DE MOV
1955 ;
1956        DEFB 0,15,31,63
1957        DEFB 127,99,127,127
1958        DEFB 0,240,248,252
1959        DEFB 254,198,254,25        4
1960        DEFB 127,115,50,16
1961        DEFB 24,8,8,0
1962        DEFB 254,206,76,8
1963        DEFB 24,16,16,0
1964 ;
1965 ;GRAFICO DE LA NAVE
1966 ;
1967 GRAFB  DEFB 0,0,1,7
1968        DEFB 63,127,112,96
1969        DEFB 24,60,255,255
1970        DEFB 255,255,0,0
1971        DEFB 0,0,128,224
1972        DEFB 252,254,14,6
1973        DEFB 0,120,68,66
1974        DEFB 66,68,120,0
1975 EXPLOS DEFB 0,59,106,85
1976        DEFB 42,53,26,13
1977        DEFB 0,16,168,88
1978        DEFB 176,88,168,84
1979        DEFB 26,53,87,45
1980        DEFB 56,16,0,0
1981        DEFB 172,84,234,86
1982        DEFB 136,4,0,0
1983 TIRO   DEFB 24,24,24,24,24
1984        DEFB 24,24,24
1985 VACIO  DEFW 0,0,0,0,0
1986        DEFW 0,0,0,0,0
1987        DEFW 0,0,0,0,0
1988        DEFW 0
1989 B1     DEFB 1,3,7,15
1990        DEFB 31,63,127,255
1991        DEFB 255,255,255,25        5
1992        DEFB 255,255,255,25        5
1993        DEFB 255,255,255,25        5
1994        DEFB 255,255,255,25        5
1995        DEFB 128,192,224,24        0
1996        DEFB 248,252,254,25        5
1997 B2     DEFB #FF,#FF,#FF,#F        F
1998        DEFB #FF,#FF,#FF,#F        F
1999        DEFB #FF,#FF,#FF,#F        F
2000        DEFB #FF,#FF,#FF,#F        F
2001        DEFB #FF,#FF,#FF,#F        F
2002        DEFB #FF,#FF,#FF,#F
2003        DEFB #FF,#FF,#FF,#F
2004        DEFB #FF,#FF,#FF,#F
2005 B3     DEFB 255,254,252,24
2006        DEFB 240,224,192,19
C2007       DEFW 0,0,0,0
2008        DEFW 0,0,0,0
2009        DEFB 255,127,63,31
2010        DEFB 15,7,3,3
2011 OVNI   DEFB 1,3,11,27
2012        DEFB 12,23,35,96
2013        DEFB 128,192,208
2014        DEFB 152,48,232,196
2015        DEFB 6
2016 ;
2017 ; VARIBLES
2018 ; DEL PROGRAMA
2019 ;
2020 POSOV  DEFB 0 ;POS OVNI
2021 OVNF   DEFB 0 ;FLAG OVNI
2022 MFLAG  DEFB 0 ; MUERTE
2023 NDISP  DEFB 0 ;NO.DISP
2024 SCORE  DEFW 0 ;PUNTOS
2025 FLGD   DEFB 0 ;FLAG DISP
2026 PYD    DEFB 0 ;POS.DISP
2027 PXD    DEFB 0 ;POS DISP
2028 CBX    DEFB 0 ;POS.BASE
2029 OLDS   DEFB 1
2030 MAYX   DEFB 0 ;MAYOR COORD
2031 MENX   DEFB 0 ;MENOR COORD
2032 MAYY   DEFB 0 ;MAYOR COORD
2033 TAM
2034        ORG TAM
2035 TX     DEFB 0 ; TAM.X.SPRI
2036 TY     DEFB 0 ;TAM.Y.SPRIT
2037 NMAR   DEFB 0 ; NO.MARC
2038 LEVEL  DEFB 1 ; NIVEL DIFI
2039 INCY   DEFB 0 ; INC DESP.
2040 INCX   DEFB 0 ; INC DESP
2041 CON    DEFB 0
2042 CON1   DEFB 0
2043 CICLOS DEFB 0
2044 CICL2  DEFB 0
2045 VIDAS  DEFB 0
```



Programa: Master Number para Amstrad

Como prometimos en tomos anteriores, aparece a continuación una versión del MASTER MIND numérico para AMSTRAD. El objetivo del juego consiste en adivinar una secuencia de números que piensa el ordenador, antes de que se nos acaben las oportunidades.

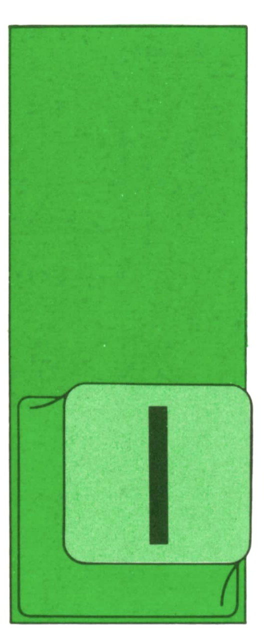
```
10 REM *****************************
20 REM ***     MASTER NUMBER     ***
30 REM *** Un programa realizado ***
40 REM ***          Por          ***
50 REM *** Carlos A. Maria Morin ***
60 REM ***                       ***
70 REM ***     (C) Ediciones     ***
80 REM ***  Siglo Cultural 1987  ***
90 REM *****************************
100 REM
110 REM ***********************************
120 REM ** INICIALIZACION Y PRESENTACION **
130 REM ***********************************
140 REM
150 MODE 0
160 LOCATE 1,13
170 PRINT "   MASTER NUMBER"
180 FOR ww=1 TO 3000
190 NEXT
200 MODE 2
210 LOCATE 33,1
220 PRINT"MASTERNUMBER"
230 INPUT "Numero de intentos:";numint
240 INPUT "Juego de carac. numericos [4..9]:";juego
250 alea1=INT(RND*juego)
260 alea2=INT(RND*juego)
270 alea3=INT(RND*juego)
280 alea4=INT(RND*juego)
290 FOR bucle=1 TO numint
300 PRINT
310 PRINT"INTENTO NUMERO";bucle
320 INPUT "!TU TURNO!";tu$
330 IF tu$="" THEN 320
340 IF LEN(tu$)<4 OR LEN(tu$)>4 THEN 320
350 FOR bu=1 TO 4
360 tes$=MID$(tu$,bu,1)
370 IF tes$<"0" OR tes$>"9" THEN 320
380 NEXT
390 tu=VAL(tu$)
400 REM
410 REM ***********************************
420 REM *** TRATAMIENTO DE VARIABLES Y ****
430 REM ****** DISGREGACION NUMERICA ******
440 REM ***********************************
450 REM
460 in=0
470 suma=0
480 kill=0
490 carac1=alea1
500 carac2=alea2
510 carac3=alea3
520 carac4=alea4
530 dis1=INT(tu/1000)
540 help1=tu-dis1*1000
550 dis2=INT(help1/100)
560 help2=help1-dis2*100
```

```
570 dis3=INT(help2/10)
580 dis4=INT(help2-dis3*10)
590 REM
600 REM ************************************
610 REM * SI ES IGUAL SUMA UNA AL CONTADOR *
620 REM ************************************
630 REM
640 IF dis1=carac1 THEN GOSUB 1070:GOSUB 860:GOSUB 1110
650 IF dis2=carac2 THEN GOSUB 1140:GOSUB 860:GOSUB 1180
660 IF dis3=carac3 THEN GOSUB 1210:GOSUB 860:GOSUB 1250
670 IF dis4=carac4 THEN GOSUB 1280:GOSUB 860:GOSUB 1320
680 REM
690 REM ************************************
700 REM ***** SI NO SON IGUALES TESEAR *****
710 REM ****** SI ESTAN MAL COLOCADAS ******
720 REM ************************************
730 REM
740 change=dis1
750 GOSUB 920
760 change=dis2
770 GOSUB 920
780 change=dis3
790 GOSUB 920
800 change=dis4
810 GOSUB 920
820 os$=""
830 GOSUB 1410
840 NEXT
850 GOTO 1590
860 kill=kill+1
870 change=10+kill
880 kill=kill+1
890 reset=10+kill
900 RETURN
910 REM
920 REM ************************************
930 REM ******** SE COMPARAN CIFRAS ********
940 REM ************************************
950 REM
960 IF change=carac1 THEN kill=kill+1:in=in+1:carac1=10+kill
970 IF change=carac2 THEN kill=kill+1:in=in+1:carac1=10+kill
980 IF change=carac3 THEN kill=kill+1:in=in+1:carac1=10+kill
990 IF change=carac4 THEN kill=kill+1:in=in+1:carac1=10+kill
1000 RETURN
1010 REM
1020 REM ************************************
1030 REM ***** SUBRUTINAS SUBORDINADAS ******
1040 REM ******* PROGRAMA PRINCIPAL *********
1050 REM ************************************
1060 REM
1070 suma=suma+1
1080 change=dis1
1090 reset=carac1
1100 RETURN
1110 dis1=change
1120 carac1=reset
1130 RETURN
1140 suma=suma+1
1150 change=dis2
1160 reset=carac2
1170 RETURN
1180 dis2=change
1190 carac2=reset
1200 RETURN
1210 suma=suma+1
1220 change=dis3
1230 reset=carac3
```

```
1240 RETURN
1250 dis3=change
1260 carac3=reset
1270 RETURN
1280 suma=suma+1
1290 change=dis4
1300 reset=carac4
1310 RETURN
1320 dis4=change
1330 carac4=reset
1340 RETURN
1350 REM
1360 REM ***********************************
1370 REM **** MENSAJES DE LOS ACIERTOS *****
1380 REM ***** PLENOS Y MAL COLOCADOS ******
1390 REM ***********************************
1400 REM
1410 PRINT"ACIERTOS PLENOS:";
1420 FOR carac1=1 TO suma
1430 os$=os$+"*"
1440 NEXT
1450 PRINT os$
1460 as$=""
1470 PRINT"CIFRAS CORRECTAS MAL COLOCADAS:";
1480 FOR carac1=1 TO_in
1490 as$=as$+"-"
1500 NEXT
1510 PRINT as$
1520 IF suma=4 THEN GOSUB 1760:GOTO 1700
1530 RETURN
1540 REM
1550 REM ***********************************
1560 REM ** MENSAJES DE ACIERTO DE NUMERO **
1570 REM ***********************************
1580 REM
1590 FOR ret=1 TO 200
1600 NEXT
1610 FOR ww=1 TO 25
1620 PRINT
1630 NEXT
1640 PRINT SPACE$(18);"Espero que la proxima vez tengas mas ingenio."
1650 PRINT
1660 PRINT SPACE$(26);"El numero era el";alea1;alea2;alea3;alea4
1670 FOR ww=1 TO 10
1680 PRINT
1690 NEXT
1700 PRINT SPACE$(22);"]Quieres intentarlo de nuevo? (S/N)"
1710 a$=UPPER$(INKEY$)
1720 IF a$<>"S" AND a$<>"N" THEN 1710
1730 IF a$="S" THEN 200
1740 PRINT"ADIOS ABUR BYE...."
1750 END
1760 PRINT
1770 PRINT"LO HAS CONSEGUIDO"
1780 PRINT"Y EN TAN SOLO";bucle;" INTENTOS.....";
1790 FOR ww=1 TO 2000:NEXT
1800 PRINT"!!!QUE FIERA!!!"
1810 PRINT
1820 RETURN
```

TECNICAS DE ANALISIS

TERMINALES DE PANTALLA (II)





Utilización

INDEPENDIENTEMENTE de la finalidad a la que se destinen los datos que se están procesando, la pantalla puede ser utilizada de varias formas en cuanto a su formato y a la distribución de los campos que en ella aparecen:

— Modo continuo. Es el modo más directo de utilización y en el que entra el ordenador, usualmente, de un modo primario cuando comienza su funcionamiento. Consiste, sencillamente, en que van apareciendo en la pantalla los datos, mensajes, preguntas, etc., en sucesivas líneas y dejando, a lo sumo, algunas líneas o espacios en blanco para mayor claridad de los textos. Excepto que se prevea lo contrario, cuando se acaba la pantalla se produce un desplazamiento hacia arriba de todo el contenido de la pantalla («roll up»), perdiéndose la primera línea que teníamos escrita y dejando una línea en blanco al final, donde se escribe la nueva línea de texto.

— Modo «patrón cerrado» o «de pantalla completa». Por este procedimiento se diseña la pantalla como una unidad distribuyendo sus campos del modo más adecuado a la utilidad a la que se destinan y se presenta de una vez al operador. Hay que disponer del software específico para realizar esta tarea, pero es enormemente cómodo disponer de toda la pantalla para presentar los datos necesarios. Por otro lado, si se hacen preguntas o se solicitan datos, saber la ubicación exacta (estable) de la respuesta dentro de la pantalla es muy útil. Se utiliza este tipo de pantalla cuando se va a hacer uso de los diversos atributos disponibles en el terminal (de los que hablaremos más adelante): vídeo inverso, parpadeo, sobreintensidad, etc.

— Ventanas. Se llaman ventanas a unas regiones de la pantalla que se aíslan del resto mediante un recuadro y se tratan como una pequeña pantalla con independencia de lo que suceda en las restantes ventanas (si las hay) o del resto de la pantalla. Dependiendo del tipo de terminal de que se trate y del software disponible, podrán «abrirse» en la pantalla más o menos ventanas. Naturalmente, puede suceder que el programa con el que se esté operando permita la utilización de cada ventana o de alguna(s) de ella(s) en modo continuo (con roll up) o en modo «pantalla completa».



Distribución de los campos

Es usual distribuir la pantalla en tres regiones: cabecera, pie y centro o núcleo del diseño.

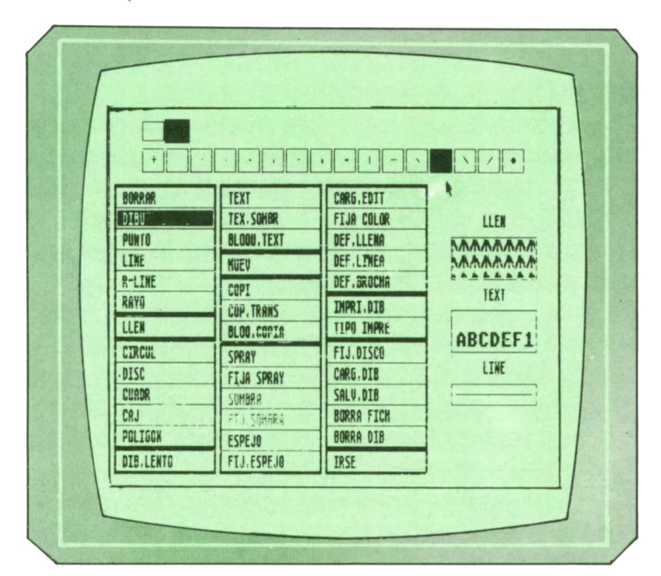
— En la cabecera se suelen poner informaciones generales sobre el proceso que se está realizando, la designación de la propia pantalla en que se opera, la fecha, etc. Además, es usual incluir en la cabecera informaciones «de ayuda»: códigos a utilizar, comandos que se pueden usar en la presente fase del proceso, etc.; a veces también datos sobre los valores permisibles en los diferentes campos de la pantalla, en general o según la situación del proceso.

— Al pie de la pantalla es normal incluir los diálogos de gobierno del proceso y los mensajes de error que se van produciendo. En ocasiones es en este área donde se incluyen las informaciones necesarias al usuario sobre el tiempo o espacio que resta para concluir la actividad en curso o sobre el resto del proceso.

— En el espacio central de la pantalla (al que se suele dedicar del 60 al 80 por ciento de la superficie total disponible) se incluyen las informaciones básicas del proceso: texto que se está preparando en un procesador de textos,

cuadro de datos de la hoja electrónica, etc. Naturalmente esta distribución descrita se refiere al formato de pantalla completa, aunque en ocasiones la zona central está preparada para que vaya haciendo «roll up» mientras que el pie y la cabecera se mantienen fijos (o se actualizan campo a campo, pero sin desplazamiento).

Por otro lado, cuando se utilizan ventanas, es usual dedicar una de ellas al control del proceso, otra a la información de errores, etc.




Pantalla de menú de un programa de gráficos.



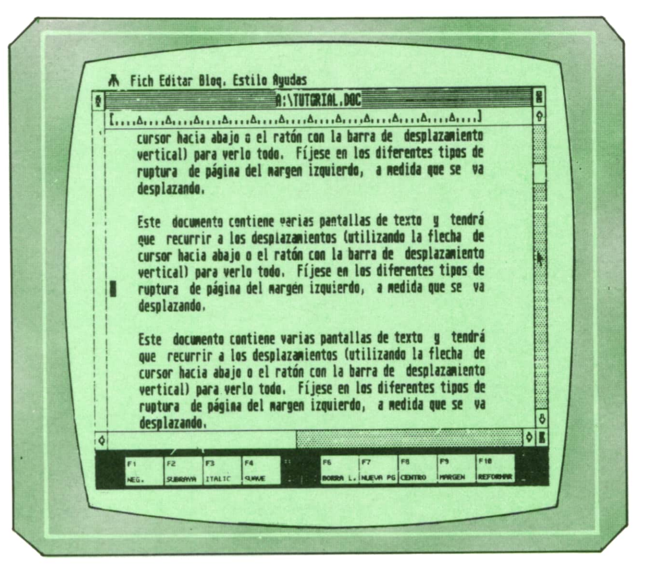
Opciones de presentación

Para la fácil identificación de los diferentes campos, realce de algunos, etc., se utilizan una serie de características de presentación visual de que suelen disponer las pantallas de los terminales. Estas características u opciones se suelen llamar «atributos» de la presentación. Los más usuales son:

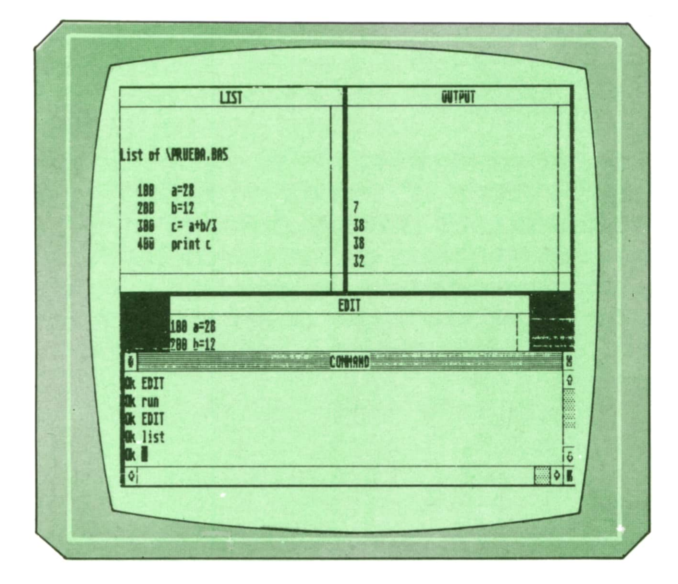
— Vídeo inverso. Consiste en cambiar los atributos de fondo y texto en una zona; si la pantalla en su conjunto está con fondo blanco y letras negras, la zona de vídeo inverso aparece con fondo negro y letras blancas, y viceversa.

— Subrayado. En la pantalla del ordenador se puede subrayar un texto, como en cualquier sistema de escritura.

— Fondo. Se puede preparar una zona con un fondo de otro nivel luminoso o color para realce de lo que se escribe sobre ella.




Editor de textos en pantalla completa con tres zonas (cabecera, pie y cuerpo central).



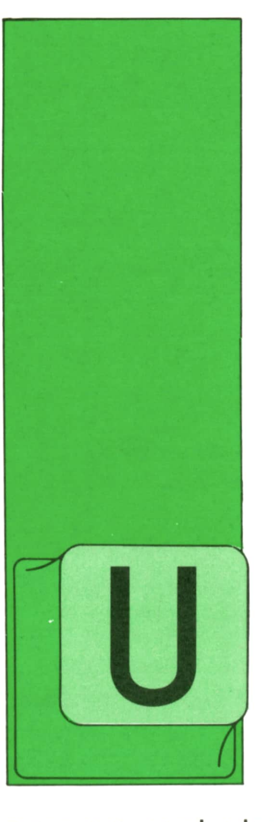

Pantalla con cuatro ventanas de un intérprete de BASIC (en cada ventana se realiza un «roll up» independiente).

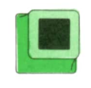
— Sobreintensidad. Normalmente también es posible escribir algunos caracteres (o todos los de una zona) con mayor intensidad luminosa.

— Parpadeo. A veces se pueden definir algunos caracteres con esta característica, para llamar la atención del usuario sobre ellos.

— No presentación. Cuando un campo (de clave o control) es confidencial, no conviene que aparezca visible en la pantalla: a medida que el usuario va escribiendo los caracteres de ese campo, el sistema los va recibiendo, pero en la pantalla no aparece nada escrito.

TECNICAS DE PROGRAMACION





Operaciones de entrada y salida

UNA de las cuestiones más importantes de la programación de ordenadores es el acceso a los dispositivos periféricos. Un programa que sólo uti!izara cálculos, pero no contuviera instrucciones de entrada y salida, no serviría de mucho, pues no podríamos pasarle datos ni enterarnos del resultado de las operaciones.

Las operaciones de entrada y salida más frecuentes en los ordenadores electrónicos pertenecen a los siguientes tipos:

— Entrada y salida de datos por la consola.
— Manejo de la pantalla.
— Impresora.
— Ficheros.
— Generación de música.
— Gráficos.
— Comunicaciones de varios ordenadores entre sí.

En este capítulo y en los sucesivos veremos con más detalle algunas de estas operaciones.



Entrada y dalida de datos por la consola

La consola (o la estación terminal, como también se llama) es uno de los elementos fundamentales del ordenador y consta de dos partes:

— El teclado.
— La pantalla.

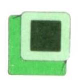
El teclado es una unidad de entrada de datos, en general muy semejante a una máquina de escribir eléctrica. Está conectado al ordenador mediante un cable y cada vez que presionamos una tecla envía por dicho cable varias señales, de las que el sistema operativo o el BIOS (siglas de «Basic Input Output System», o «sistema básico de control de la entrada y la salida») puede deducir la tecla que se ha presionado y convertir esta información en la representación interna del carácter correspondiente. Desde nuestro punto de vista de programadores de lenguajes de alto nivel, en general no tendremos que preocuparnos de la conversión de las señales eléctricas recibidas desde el teclado en los caracteres correspondientes. Bastará con tratar directamente con éstos.

La pantalla es, primordialmente, una unidad de salida de datos, y para todos los efectos puede considerarse compuesta por una serie de líneas y columnas que forman una cuadrícula, en cada uno de cuyos cuadros puede colocarse un solo carácter. El número de líneas y columnas de las pantallas varía según el ordenador, y a veces en una misma máquina es posible cambiar dichos números y trabajar con varios formatos de pantalla. Una de las disposiciones más comunes es la de 25 líneas y 40 u 80 columnas.

El teclado no puede convertirse nunca en unidad de salida de datos, pero la pantalla sí puede utilizarse a veces como unidad de entrada, si se la dota de dispositivos especiales, como lápiz electrónico, detector de la posición del dedo, etc. En este curso de técnicas de programación no haremos más que mencionar esta posibilidad.

Las operaciones fundamentales que pueden realizarse con la consola son las siguientes:

— Lectura de una línea desde el teclado.
— Escritura de una línea en la pantalla.
— Lectura de un carácter desde el teclado.
— Escritura de uno o varios caracteres (que no forman línea) en la pantalla.
Los dos primeros son más básicos y los estudiaremos en primer lugar.



Lectura de una línea desde el teclado

Es frecuente (y lo hemos visto varias veces en los ejemplos de programas propuestos en los capítulos anteriores) que deseemos introducir datos en el ordenador, para que nuestros programas trabajen con ellos. La forma más sencilla de conseguirlo es proporcionárselos a través del teclado, escribiendo una línea de caracteres, que terminará siempre con el carácter de salto de línea, que a veces se llama ENTER. Lo normal es que, al mismo tiempo que escribimos los caracteres de esta línea, podamos ver cómo van apareciendo en la pantalla.

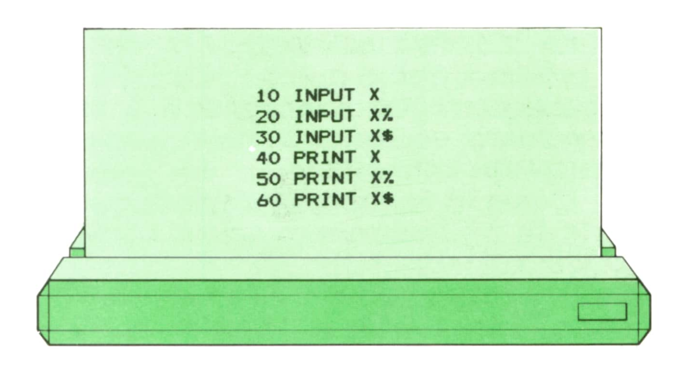
En el lenguaje BASIC la instrucción que permite leer una línea de la pantalla comienza por la palabra reservada INPUT. En su forma más sencilla, esta instrucción tiene el siguiente aspecto:

INPUT variable

que nos dice que el dato leído desde el teclado debe ser asignado a la variable indicada. El funcionamiento de la instrucción INPUT es, por tanto, el siguiente:

1. El teclado queda disponible para escribir.
2. La persona sentada ante la consola teclea el dato pedido.
3. El dato es convertido al tipo interno de la variable.
4. El resultado de la conversión es asignado a la variable.

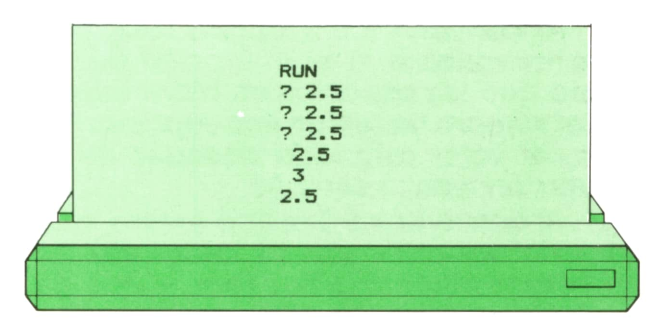
Veamos, como ejemplo, el programa siguiente:



```
10 INPUT X
20 INPUT X%
30 INPUT X$
40 PRINT X
50 PRINT X%
60 PRINT X$
```

Este programa realiza las siguientes operaciones:
1. La primera instrucción lee un dato, lo convierte a la representación interna apropiada a la variable x (un número real en punto flotante) y se lo asigna a la variable x.
2. La segunda instrucción lee un dato, lo convierte a la representación interna apropiada a la variable x% (un número entero) y se lo asigna a la variable x%.
3. La tercera instrucción lee un dato, lo convierte a la representación interna apropiada a la variable x$ (una cadena de caracteres; por tanto, no hay conversión) y se lo asigna a la variable x$.
4. Las restantes instrucciones escriben en la pantalla los valores respectivos de x, x% y x$, para que podamos ver los valores que recibieron dichas variables.

Veamos cómo se ejecuta el programa anterior:



```
RUN
? 2.5
? 2.5
? 2.5
 2.5
 3
2.5
```

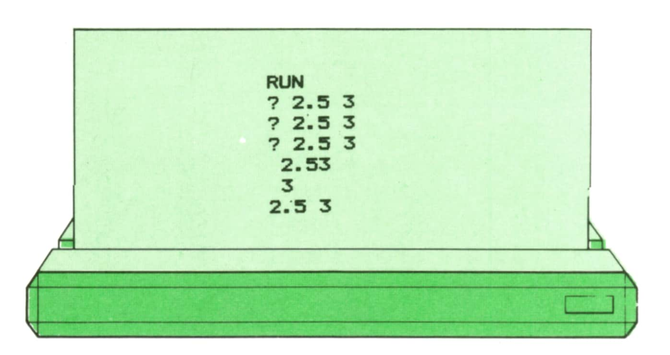
Observaremos que, cuando se ejecuta la instrucción INPUT, el intérprete de BASIC escribe un signo de interrogación para indicarnos que el programa está esperando datos. Nosotros responderemos en todos los casos tecleando los tres caracteres «2.5» y presionando la tecla ENTER.

Del análisis de los resultados podemos deducir lo siguiente:
1. Como la variable x tiene tipo interno real, el valor asignado fue el número 2.5. Con este número podríamos ahora hacer operaciones aritméticas, como, por ejemplo, sumarle una unidad.
2. Como la variable x% tiene tipo interno entero, el intérprete ha convertido

el valor 2.5, redondeándolo al entero más próximo, por lo que a la variable se le ha asignado un valor igual a 3. Con este número podemos también realizar operaciones aritméticas.

3. Como la variable x$ tiene tipo literal, no se ha realizado ninguna conversión y el valor asignado ha sido la cadena de tres caracteres «2.5». Con este valor no pueden realizarse operaciones aritméticas, pues no se trata de un número, sino de una cadena de caracteres.

¿Qué sucederá si, en lugar de proporcionar un solo dato cuando se nos pide, respondemos con dos? Veamos un ejemplo:



Veamos cómo se interpretan estos resultados:

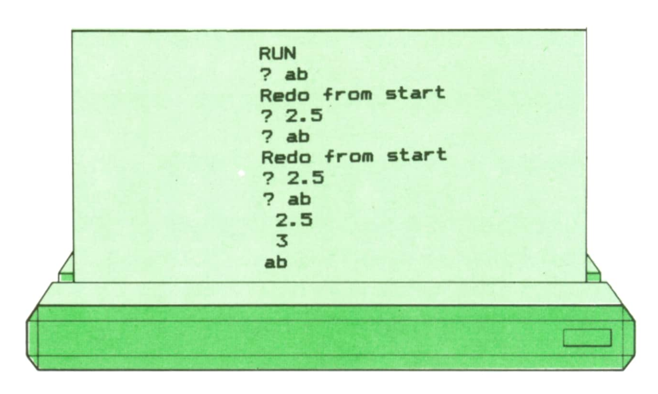
1. Al convertir 2.5 3 al tipo real, que exige la variable x, el intérprete ha supuesto que los espacios en blanco dentro del número no tenían importancia. Por tanto, el valor asignado después de la conversión resultó ser 2.53.

2. Al convertir 2.5 3 al tipo entero, que exige la variable x%, el intérprete ha convertido primero al tipo real, lo que da el mismo resultado que en el caso anterior (2.53) y después ha redondeado este número al entero más próximo, por lo que el valor asignado a x% fue 3.

3. Como la variable x$ tiene tipo literal, no se ha realizado ninguna conversión y el valor asignado ha sido la cadena de cinco caracteres que hemos introducido («2.5 3»). Obsérvese que el espacio en blanco cuenta como un carácter y aparece dentro del valor de x$ en la misma posición en que lo escribimos en el teclado.

¿Qué ocurre si a una variable numérica le asignamos algo que no se puede convertir en un número? Veámoslo:

1. El intérprete no nos deja responder a la petición de datos para la variable x

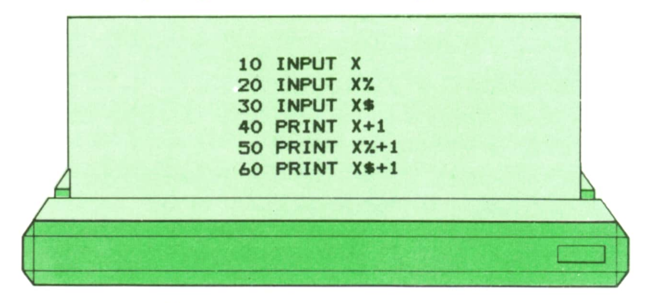


con la cadena de caracteres «ab», pues esta cadena no se puede convertir en un valor numérico. Por tanto, responde con una mensaje de error («redo from start», que quiere decir «vuélvalo a hacer desde el principio») y pide de nuevo un valor para la variable x. Esta vez le damos uno válido (2.5).

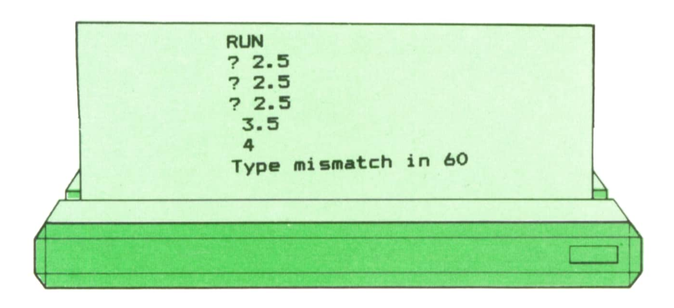
2. El intérprete no nos deja responder a la petición de datos para la variable x% con la cadena de caracteres «ab», pues esta cadena no se puede convertir en un valor numérico. Por tanto, responde con el mismo mensaje de error y pide de nuevo un valor para la variable x%. Esta vez le damos uno válido (2.5), que será convertido a entero, redondeándolo a 3.

3. No hay ningún problema para asignarle a x$ un valor que no se puede convertir en numérico, pues x$ tiene tipo literal. Por tanto, en este caso el intérprete acepta el valor que le hemos dado («ab»).

Vamos a comprobar que con los valores numéricos asignados a las variables literales no se pueden realizar operaciones. Es decir: que 2.5 asignado a la variable x es distinto de 2.5 asignado a la variable x$. Para ello modificaremos nuestro programa de la siguiente forma:


```
10 INPUT X
20 INPUT X%
30 INPUT X$
40 PRINT X+1
50 PRINT X%+1
60 PRINT X$+1
```

donde lo único que ha cambiado es que, en lugar de escribir los valores de las variables, escribimos el resultado de sumar una unidad a cada una de ellas. Veamos cómo se ejecuta este nuevo programa:



Veamos lo que quiere decir este resultado:

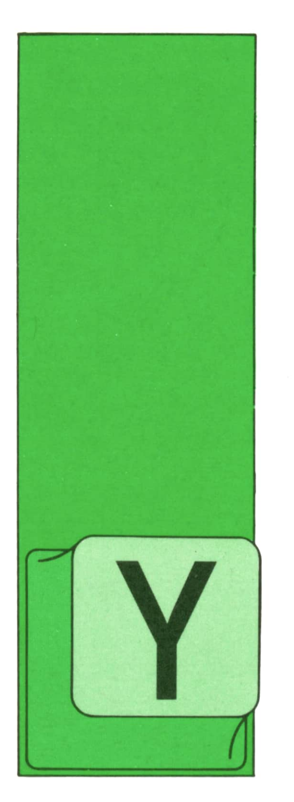
1. El intérprete ha asignado el valor 2.5 a la variable x, como en los casos anteriores. Por tanto, al escribir x+1, el valor obtenido será 3.5.

2. El intérprete ha asignado el valor 3 a la variable x%, como en los casos anteriores. Por tanto, al escribir x%+1, el valor obtenido será 4.

3. El intérprete ha asignado el valor literal «2.5» a la variable x%, como en los casos anteriores. Por tanto, al intentar escribir x$+1, obtendremos un mensaje de error («type mismatch in 60», que significa «tipo incorrecto en la línea 60») y la operación no se realiza.

LOGO

RECURSIVIDAD



Ya sabemos que al definir un procedimiento para enseñarle a la tortuga a hacer cosas nuevas no sólo podemos utilizar comandos, sino que también se pueden poner los nombres de otros procedimientos. Pues bien, se dice que un procedimiento es *recursivo* cuando se usa a sí mismo, es decir, que dentro de su definición escribimos su propio nombre. De esta manera, repetiremos muchas veces el conjunto de órdenes que componen ese procedimiento.

Por ejemplo, supongamos que quisiéramos tener un procedimiento para que la tortuga estuviera andando continuamente. Para ello tendríamos que decirle que avanzara un paso infinitas veces:

```
? PARA ANDAR
> AV 1
> ANDAR
> FIN
```

pero esto es imposible. En cambio, si ponemos

```
? PARA ANDAR
> AV 1
> AV 1
> AV 1
> FIN
```

la tortuga avanza un paso, busca lo que significa ANDAR, avanza un paso, busca lo que significa ANDAR, avanza un paso..., es decir, no pararía de avanzar por la pantalla.

Por tanto, vemos que, de momento, si decimos a la tortuga que ejecute un procedimiento recursivo nunca termina de hacerlos porque siempre vuelve a empezar el mismo procedimiento.

Para que la tortuga pare de ejecutarlo tendremos que dar una o varias teclas del teclado (normalmente, la tecla BREAK):



Figura 1

Al pulsar esta tecla, la tortuga nos pone un mensaje en la pantalla diciéndonos que ha parado la ejecución:

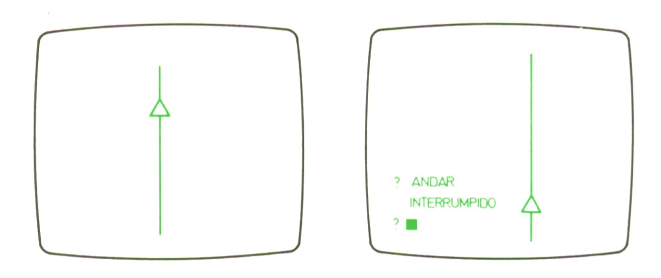
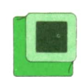


Figura 2



Algunos procedimientos recursivos

Vamos a dibujar una espiral cuadrada mediante un procedimiento recursivo. Nos basta con poner:

```
? PARA ESPIRAL C: LARGO
> OT
> AV :LARGO
> GD 90
> HAZ "LARGO: LARGO + 3
> ESPIRALC :LARGO
> FIN
```

Si ahora escribimos

```
ESPIRALC 20
```

la tortuga empezaría pintando el primer lado de la espiral avanzando 20 y seguiría continuamente girando 90 y pintando el resto de los lados incrementando de 3 en 3 hasta que pulsáramos la tecla de parada.

Si queremos tener un procedimiento que sirva para dibujar cualquier espiral (triangular, cuadrada, pentagonal...) tenemos que usar el siguiente procedimiento:

```
? PARA ESPIRAL :LARGO :LADOS :PASO
> OT
> AV :LARGO
> GD 360 / :LADOS
> HAZ "LARGO :LARGO + :PASO
> ESPIRAL :LARGO :LADOS :PASO
> FIN
```

sabiendo que en LARGO guardamos el valor de cada lado de la espiral, en LADOS el número de lados de la espiral y en PASO la cantidad que sumamos para ir aumentando el tamaño de los lados. Un ejemplo de ejecución puede ser:

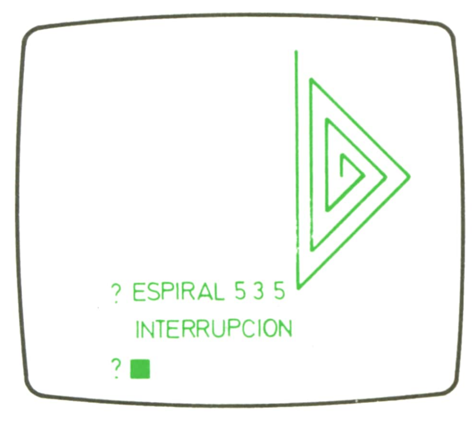


Figura 3

Usando también un procedimiento recursivo podemos pintar, por ejemplo, la siguiente figura:

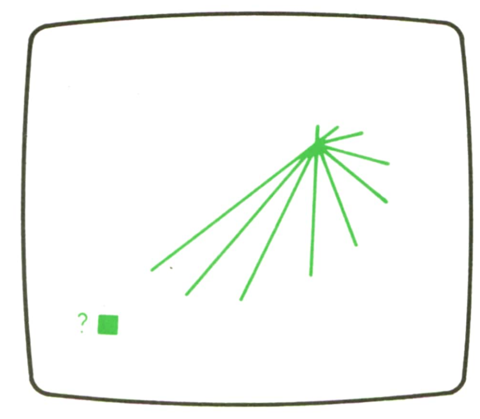


Figura 4

haciendo que cada raya sea de un color aleatorio.

El procedimiento correspondiente es el siguiente:

```
? PARA RAYAS :LONG :ANG :PASO
> OT
> PONCL AZAR 16
> AV :LONG
> RE: LONG
> GD :ANG
> HAZ "LONG :LONG + :PASO
> RAYAS :LONG :ANG :PASO
> FIN
```

Como vemos, en general, al escribir procedimientos recursivos que usan variables, antes de volver a llamar al procedimiento escribimos un comando HAZ. Para evitar esto podemos variar directamente el valor de las variables al utilizar el propio procedimiento de nuevo. Por ejemplo, el procedimiento RAYAS se puede poner así:

```
? PARA RAYAS :LONG :ANG :PASO
> OT
> PONCL AZAR 16
> AV :LONG
> RE :LONG
> GD :ANG
> RAYAS :LONG + :PASO :ANG :PASO
> FIN
```

Por último, para pintar esta figura

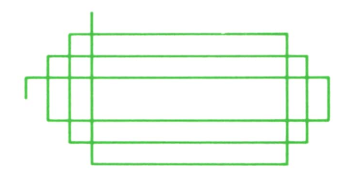


Figura 5

en la que las rayas verticales van siendo más grandes y las horizontales más pequeñas, se puede usar este procedimiento:

```
? PARA FIGURA :LADOV :PASOV :LA-
DOH :PASOH
> OT
> PONCL 2
> AV :LADOV GD 90
> PONCL 8
> AV :LADOH GD 90
> FIGURA :LADOV + :PASOV :PASOV
:LADOH - :PASOH :PASOH
> FIN
```

siendo LADOV y LADOH la longitud de las rayas verticales y horizontales, respectivamente, PASOV el incremento en dirección vertical y PASOH el decremento en dirección horizontal.



Condiciones

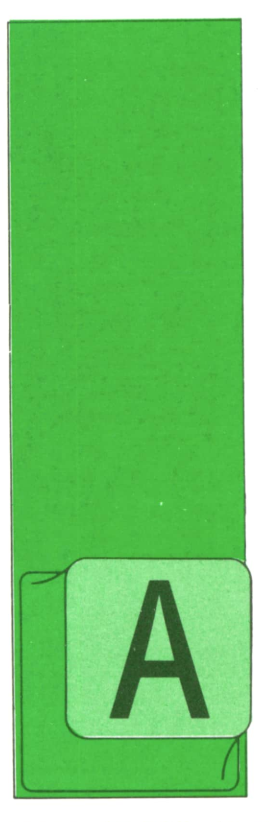
Una *condición* es algo que se tiene que cumplir para realizar una cosa. Por ejemplo, nosotros solemos decir: "Si llueve me llevaré el paraguas." Si la condición es CIERTA (llueve), se realiza la acción, mientras que si es FALSA (no llueve), no se hace.

Existen varias maneras de escribir una condición mediante la utilización de *operadores de relación* y *operadores lógicos*. La tortuga se encarga de realizar las operaciones correspondientes y de calcular un resultado. En este caso no es un número, sino CIERTO, si la condición es verdad, o FALSO, si la condición es mentira.

Como siempre, con este resultado tenemos que hacer algo. Podemos escribirlo pero esto no es demasiado útil. Donde realmente se va a usar más va a ser acompañando a un comando que veremos más adelante.

PASCAL

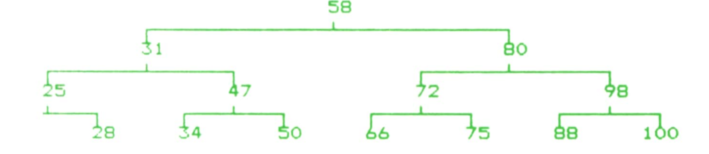
ARBOLES DE BUSQUEDA



A menudo se desea seleccionar los elementos de un árbol en función del contenido de un determinado campo (el primer apellido si son fichas de personas, o el número de DNI...); a este campo se le suele denominar campo "clave". Pues bien, se dice que un árbol binario es "de búsqueda" cuando, para cualquier nodo, se cumple que todos los nodos de su subárbol izquierdo tienen una clave que se encuentra por delante de la propia y todos los nodos de su subárbol derecho tienen una clave que se encuentra por detrás.

Al decir "por delante" o "por detrás" estamos dando por supuesto que hay un criterio de clasificación de claves; por ejemplo, si el campo clave fuese el apellido, "por delante" podría significar por delante alfabéticamente.

Veamos un ejemplo de árbol de búsqueda con claves numéricas en que "por delante" signifique "con clave menor" (representaremos sólo el campo clave):



Para buscar el elemento 34, por ejemplo, el proceso sería el siguiente:

Primero observaríamos la raíz del árbol para ver si es el elemento 34; como es el 58 y el árbol es de búsqueda, caso de que existiese el 34; debería encontrarse en el subárbol izquierdo.

Para buscar en éste, el proceso sería similar: observaríamos su raíz y, como ésta es el elemento 31, llegaríamos a la conclusión de que el elemento 34 debería encontrarse en su subárbol derecho, es decir, en el encabezado por el elemento 47.

De la comparación con este último deduciríamos que el elemento 34 debería encontrarse a su vez en su subárbol izquierdo donde, por fin, lo encontraríamos.

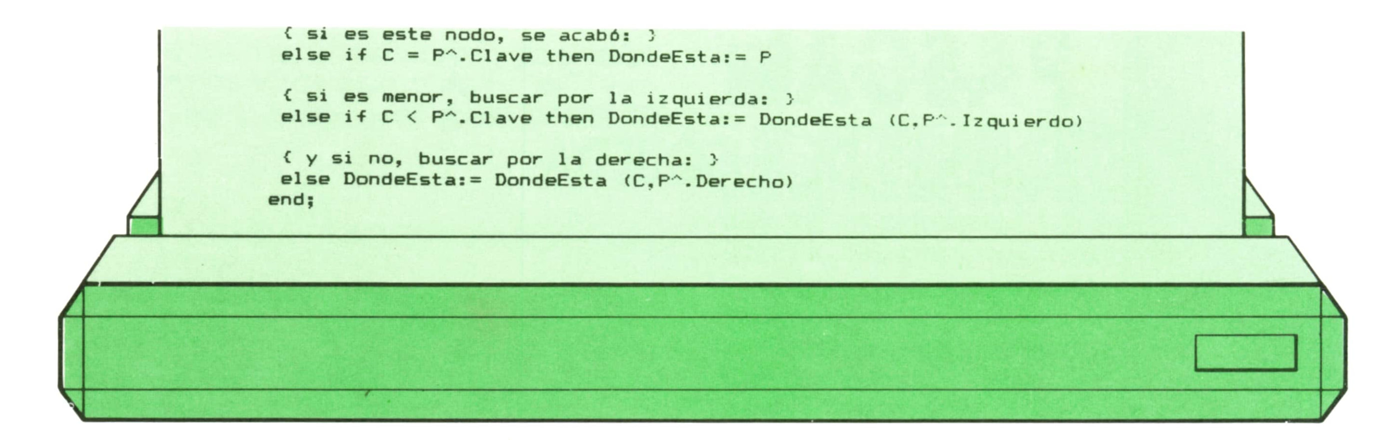
Si, tras esta secuencia de búsqueda, llegásemos a un nodo sin subárboles por donde avanzar, la conclusión sería que el elemento no está en el árbol (imaginemos que buscamos el elemento 35, por ejemplo).

Resulta claro que el procedimiento de búsqueda se puede expresar de manera recursiva muy fácilmente. Una función tal que nos devuelva un puntero al nodo buscado, o bien el valor NIL si no existe podría ser:

```
function DondeEsta (C: Clave_t; P: Puntero_t): Puntero_t;

(* Busca en el árbol apuntado por P el elemento con clave C *)
(* Devuelve un puntero al nodo si lo encuentra, o bien NIL. *)

 begin
  { si ya no hay dónde mirar, devuelve NIL: }
  if P = nil then DondeEsta:= nil
```



```
  { si es este nodo, se acabó: }
  else if C = P^.Clave then DondeEsta:= P

  { si es menor, buscar por la izquierda: }
  else if C < P^.Clave then DondeEsta:= DondeEsta (C,P^.Izquierdo)

  { y si no, buscar por la derecha: }
  else DondeEsta:= DondeEsta (C,P^.Derecho)
end;
```

Sin embargo, éste es un caso en que la solución interativa es también sencilla y, por tanto, preferible:

```
function DondeEsta (C: Clave_t; P: Puntero_t): Puntero_t;

(* Busca en el árbol apuntado por P el elemento con clave C *)
(* Devuelve un puntero al nodo si lo encuentra, o bien NIL. *)

  var EsEste: boolean;
 begin
  EsEste:= false;

  while (P <> nil) and not EsEste do
    if C = P^.Clave then EsEste:= true
    else if C < P^.Clave then P:= P^.Izquierdo
    else P:= P^.Derecho;

  DondeEsta:= P
 end;
```

Como recordará el lector, este procedimiento es casi idéntico al de búsqueda de un elemento en una lista lineal, sólo que con la posibilidad de escoger entre dos caminos distintos en cada iteración.

En el árbol de búsqueda del ejemplo, para encontrar el elemento 34 habríamos necesitado realizar cuatro comparaciones, las mismas que para cualquier otro elemento de su mismo nivel; para los elementos del nivel anterior habrían sido necesarias tres, etc. En definitiva, tendríamos que el número medio de comparaciones necesario para encontrar los elementos del árbol sería de 3,27, más o menos; además, el número necesario para llegar a la conclusión de que un elemento no se encuentra en el árbol sería de 4.

En una lista lineal ordenada con el mismo número de elementos, en promedio harían falta 8 comparaciones.

En un árbol de nivel 20, por ejemplo, podríamos tener más de un millón de elementos, y en el peor de los casos harían falta 20 comparaciones para dar por terminada una búsqueda; en una lista con una cantidad de elementos semejante haría falta, sin embargo, un número medio de comparaciones del orden de medio millón.

En una tabla ordenada de datos, mediante el procedimiento denominado "búsqueda dicotómica", se puede encontrar un elemento dado con un número de operaciones similar al de un árbol binario de búsqueda, pero, como ya se comentó en su momento, esa ventaja sobre las listas se pierde a la hora de añadir nuevos elementos si se quiere que la estructura permanezca ordenada, pues obliga cada vez a correr de sitio parte de los elementos.

El árbol reúne las mejores característi-

cas tanto para la búsqueda como para la inserción de elementos.

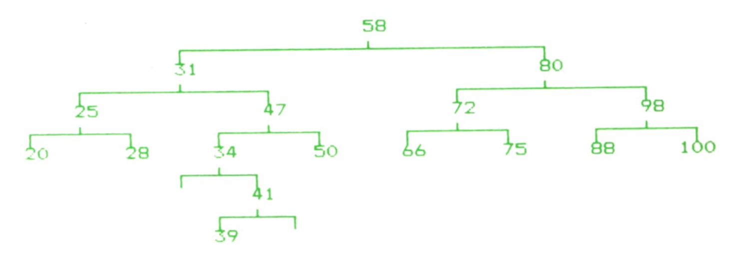
 Inserción de elementos

En principio, las operaciones necesarias para insertar un nuevo elemento en el sitio correcto de un árbol de búsqueda parecen sencillas.

Supongamos que en el árbol del ejemplo queremos insertar el elemento 41. Para empezar, lo compararíamos con el nodo raíz y, al estar antes que éste, llegaríamos a la conclusión de que el nuevo elemento habría de ubicarse en su subárbol izquierdo; pasaríamos entonces a compararlo con el nodo 31 y, como éste debería encontrarse ante de 41, pasaríamos a investigar por su subárbol derecho. De esta manera llegaríamos al nodo 34; aquí ya no se puede avanzar más, por lo que la búsqueda se ha terminado: el nuevo elemento debe ubicarse como descendiente directo y, al estar detrás, debe ser descendiente derecho.

Si ahora quisiéramos añadir el elemento 39, por razonamientos parecidos llegaríamos a la conclusión de que su lugar en el árbol es el de descendiente directo izquierdo del nodo 41.

El árbol, tras la inclusión de estos dos elementos, quedaría así:



Queda claro que el árbol sigue siendo de búsqueda.

El procedimiento de inserción, en principio, podría ser similar en la fase inicial de búsqueda a los que ya hemos visto; sin embargo, al llevar a la práctica la inserción del nuevo elemento hay que modificar uno de los dos punteros del nodo al que va a ser enlazado.

Aunque el método iterativo se puede utilizar aquí también, se complica ligeramente, pues hay que guardar en todo momento un puntero al nodo antecesor a aquél en que nos encontremos, para poder así modificarlo en su caso. El método recursivo es fácilmente adaptable:

Nótese el paso del puntero "por nombre" para así poder modificar la variable que aparecería en la lista de parámetros en el momento de la llamada. Para insertar un elemento con clave 77, si el puntero que apunta al nodo raíz se llamase Raiz, haríamos:

```
InsertarEn (Raiz, 77);
```

```
procedure InsertarEn (var P: Puntero_t; C: Clave_t);

(* Añade al árbol apuntado por P un nodo con clave C *)

 begin
  if P = nil then   (* hemos llegado a un extremo *)
      begin
       new (P);   (* P pasa a apuntar a una nueva ficha *)
       with P do
          begin
           Clave:= C;
           (* otros campos... *)
           Izquierdo:= nil;
           Derecho  := nil
          end
      end

  else if C < P^.Clave then InsertarEn (P^.Izquierdo,C)

  else if C > P^.Clave then InsertarEn (P^.Derecho,C)

  else (* C es igual a P^.Clave, o sea, que ya estaba...*)
end;
```

Si la nueva ficha ya estuviera preparada, tras ejecutarse new (P) bastaría con asignarla a P^. Por otra parte, lo que ha de hacerse en caso de que la ficha que se intenta insertar se encuentre ya en el árbol depende de cada programa, como veremos próximamente con un ejemplo.

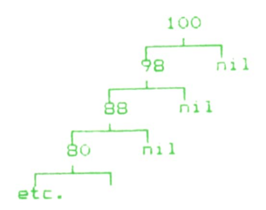


Equilibrio

Antes de empezar a crear un árbol, la variable Raiz tendrá el valor NIL y, tras la primera llamada al procedimiento InsertarEn, pasará a apuntar al que ya para siempre será el nodo raíz.

Supongamos que empezamos a insertar los nodos del árbol del ejemplo en el siguiente orden: 100, 98, 88, 80... El nodo raíz tendrá la clave 100; posteriormente, el 98 quedará, lógicamente, en su subárbol izquierdo; el 88, a su vez, en el izquierdo del 98...

Está claro que acabaríamos con un árbol que, en la práctica, sería como una lista:



En otras palabras, si, por casualidad, uno de los primeros elementos en ser insertados tiene una clave con valor muy extremo, la mayoría de los elementos que añadamos con posterioridad descenderán por el mismo subárbol.

Se denomina árbol perfectamente equilibrado a aquél en que, para cualquier nodo, se cumple que el número de nodos de su subárbol derecho y el número de nodos de su subárbol izquierdo difieren en, a lo sumo, una unidad; el primer árbol de ejemplo sería, pues, un árbol perfectamente equilibrado.

Está claro que, a medida que va desapareciendo el equilibrio de un árbol, van desapareciendo sus ventajas, hasta llegarse al caso del último ejemplo.

Aunque existen métodos para "reequilibrar" e, incluso, otros tipos de árbol con desequilibrio limitado gracias a su especial estructura, su tratamiento desborda el alcance de esta obra.



Borrado de un elemento

La operación de borrado de un elemento no es tan sencilla como la de inserción. Se pueden diferenciar dos casos:

1. El nodo a eliminar tiene, a lo sumo, descendientes por uno de sus subárboles.
2. El nodo a eliminar tiene descendientes por ambos subárboles.

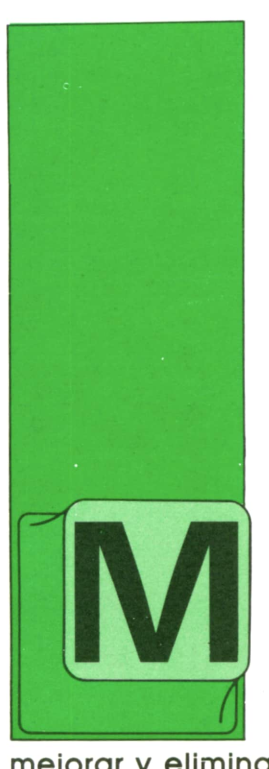
En el caso 1, la operación es muy sencilla: si carece de descendientes, basta con asignar el valor NIL al puntero de su antecesor, mientras que si sólo los tiene por un subárbol, basta con hacer que su antecesor pase a apuntar a su único descendiente inmediato (o sea, quitarlo de enmedio).

En el caso 2 no es tan sencillo: del antecesor sólo queda libre un puntero, pero hay dos subárboles a enganchar. Una solución consiste en buscar entre todos los nodos contenidos en su subárbol izquierdo el que, según el criterio de ordenación, se encuentre inmediatamente antes y ponerlo en su lugar; la otra consiste en reemplazarlo por el nodo de su árbol derecho inmediatamente posterior. Cualquiera de las dos soluciones es válida.

Para encontrar el nodo inmediatamente anterior basta con, empezando por el descendiente inmediato izquierdo, avanzar todo lo que se pueda por ramas derechas y, para el inmediatamente posterior, basta con avanzar desde el descendiente inmediato derecho por las ramas izquierdas. Tras sustituir el contenido del nodo a eliminar por el del nodo escogido, habrá que borrar este último; sea cual sea el nodo que hayamos escogido, está claro que carece por lo menos de un descendiente, pues si no, habríamos seguido avanzando, por lo que su borrado no presentará problemas.

OTROS LENGUAJES

(MODULA-2 1)



MODULA-2 es un lenguaje de programación estructurado y modular, de esta última característica proviene el nombre. Fue diseñado en 1977 por Niklaus Wirth para mejorar y eliminar los fallos de sus anteriores lenguajes de programación: MODULA y el ampliamente conocido PASCAL. Con respecto a este último posee las siguientes mejoras:

— Posibilidad de desarrollar programas en módulos y permitir la compilación separada de cada uno de ellos.

— Posibilidades de multiprogramación (corrutinas); esto es, varios programas, o procesos, ejecutándose simultáneamente.

— Facilidad para acceder a los recursos hardware del ordenador, mediante instrucciones de bajo nivel, lo que antes sólo se podía realizar en lenguaje máquina.

Todos los lenguajes de programación más modernos, como el ADA y en menor manera el C, incluyen la posibilidad de desarrollar programas en módulos.

Al realizar programas muy grandes, o complicados, éstos se suele subdividir en un cierto número de bloques lógicos, encargando al diseño de cada uno de ellos a un programador diferente. Al poco de comenzar el trabajo cada programador comienza a realizar su tarea con su propio estilo, separándose ligeramente de las especificaciones iniciales. Por otra parte, pronto se acaban los nombres para las variables, que deben indicar la función de la variable para simplificar el trabajo al depurar el programa. Para evitar todos estos problemas cada programador realizará un módulo.

En cada parte se definen el módulo de definición y el de implementación. En el primero se indica qué es lo que se puede conocer en el exterior del módulo, lo que se exporta, qué funciones necesita de otros módulos, lo que se importa. En el módulo de implementación se escribe el programa propiamente dicho, pero dentro de este podemos realizar todo lo que deseemos, con los nombres que queramos que en el programa sólo se conocerá lo que hayamos definido como exportable.

Como módulo se entiende una colección de declaraciones y una secuencia de sentencias. Comienza con MODULE y finaliza con END. La cabecera contiene los identificadores del módulo, y posiblemente las listas de los objetos que importa de otros módulos, lista de importación, y la lista de los objetos que exporta para que sean accesibles por el resto de los módulos, lista de exporación. Así que un módulo esconde todos los objetos que el programador quiere que permanezcan locales, sólo permitiendo que sean accesibles los que desee.

Los objetos locales a un módulo tendrán el mismo nivel que en un programa cualquiera; pueden considerarse locales a ciertos procedimientos del módulo, o globales a 0 uno entero.

Al final de todas las declaraciones de procedimientos del módulo viene el cuerpo del módulo, que se ejecuta cuando los procedimientos a los cuales el módulo es local son referenciadas. Si muchos módulos son declarados, entonces estas sentencias son ejecutadas en la misma secuencia en que se referencien los módulos.

La utilidad de estos cuerpos del módulo son la inicialización de las variables locales al módulo.

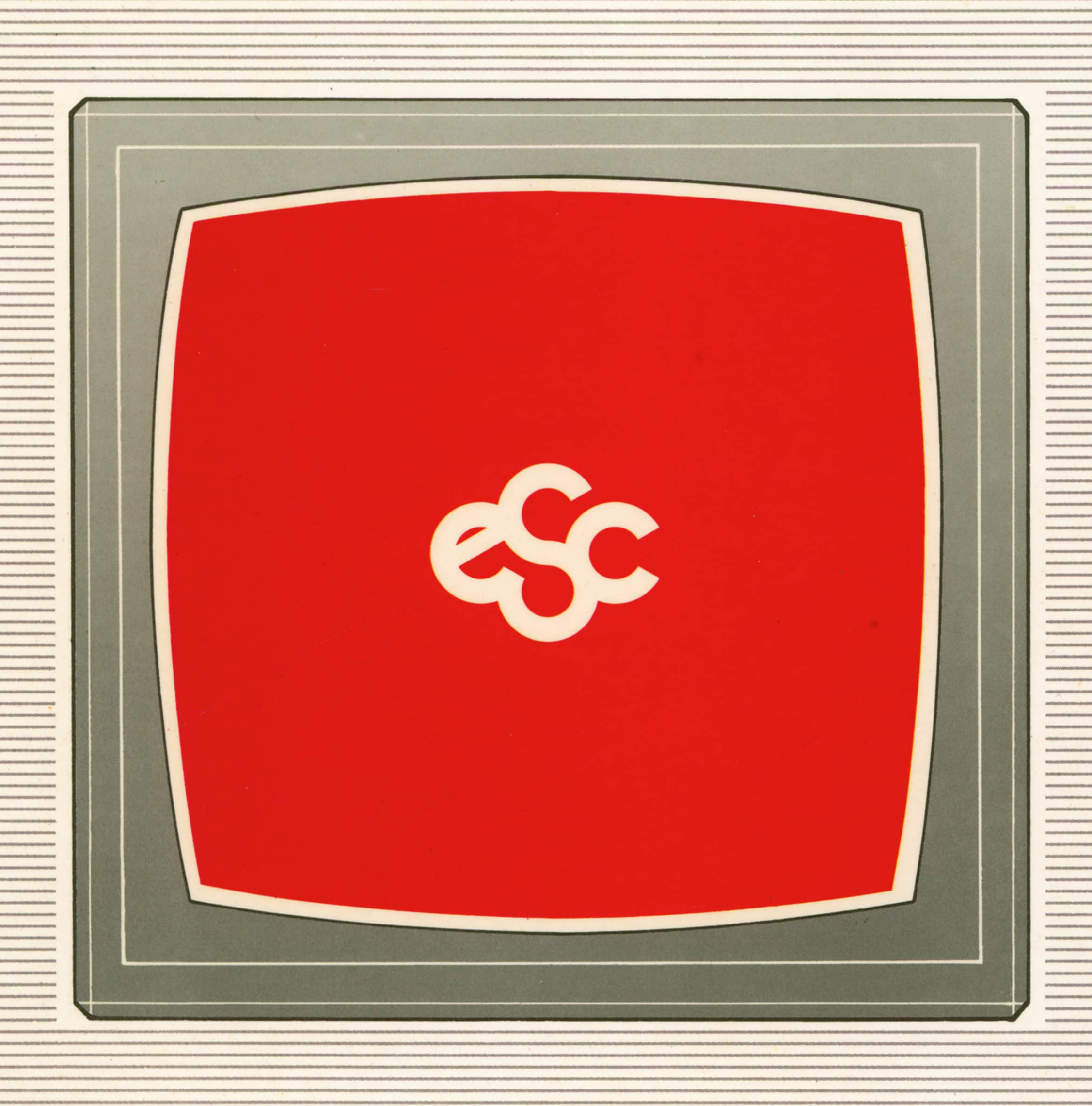
Si un identificador está en una lista de exportación supone que no hay ningún objeto con el mismo nombre en otro módulo usado. Si existiera otro deberíamos añadir la palabra clave QUALIFIED; en este caso para referenciar los objetos con esta propiedad debemos colocar el nombre del módulo del que se importan seguido de un punto y el nombre del objeto. De esta forma se evitan posibles colisiones entre objetos de diferentes módulos, que presumiblemente denotarán diferentes objetos.

Un módulo puede tener muchas listas de importación. Si éstas comienzan con FROM y el nombre del módulo del que se desea importar eliminarán la propiedad de identificador cualificado (QUALIFIED), por lo que podrán ser utilizados como identificadores corrientes.

Existen dos tipos de módulos, los de definición (DEFINITION MODULE), en los que se declaran los objetos que desean que sean conocidos por el resto de los módulos; y el módulo de implementación (IMPLEMENTATION MODULE), en el que viene el programa propiamente dicho. La razón de esta separación es que quien quiera utilizar dicho módulo sólo deberá conocer el módulo de definición, sin importarle de qué manera está implementado, utilizándolo como una caja negra que hace lo que queremos, pero que no conocemos cómo funciona.

Veamos un ejemplo de utilización de módulo:

```
MODULE MCD; (* CALCULO DEL MAXIMO COMUN DIVISOR *)
  FORM InOut IMPORT ReadCard,WriteCard;
  VAR
      X,Y : CARDINAL;
  BEGIN
    ReadCard (X);
    ReadCard (Y);
    WHILE X = Y DO
         IF X > Y THEN
               X := X - Y;
         ELSE
               Y := Y - X
         END
   END;
   WriteCard (X,6);
END MCD.
```



▼ EDICIONES ▼ SIGLO ▼ CULTURAL ▼